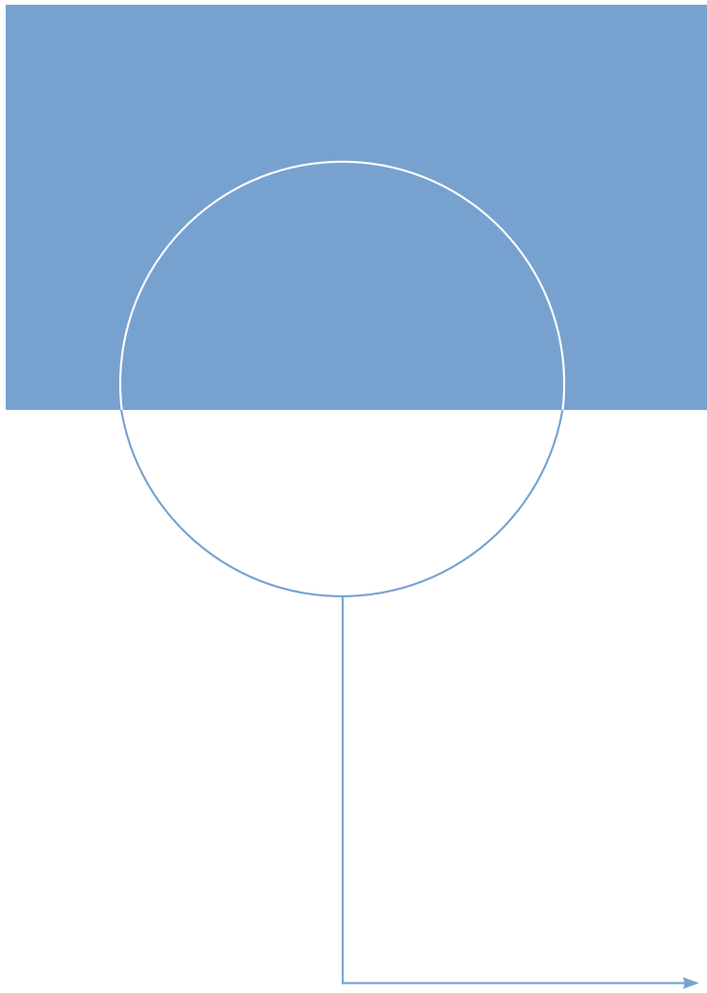


Master's thesis

2022

Joseph Vincent Broussard

NTNU
Norwegian University of
Science and Technology
Faculty of Natural Sciences
Department of Physics

Joseph Vincent Broussard

Analyzing and improving open-source template matching for orientation mapping based on SPED data

June 2022

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Analyzing and improving open-source template matching for orientation mapping based on SPED data

Joseph Vincent Broussard

Master of Science in Physics
Submission date: June 2022
Supervisor: Antonius T. J. van Helvoort
Co-supervisor: Håkon Wiik Ånes

Norwegian University of Science and Technology
Department of Physics

Abstract

Science has become and continues to be more reliant on fast, accurate, precise and understandable computational methods. Most scientific communities have been gravitating towards open source python packages, as python is more generally readable and understandable than most compiled languages, and electron microscopy is no different. With the wealth of open source packages available comes a necessity to test the limitations and efficacy of these packages. Here the python package Pyxem, built upon the Hyperspy library, is of interest due to recent developments in fast template matching of scanning precession electron diffraction data. In template matching an experimental electron diffraction pattern is compared to a bank of simulated templates, for example for different orientations of the candidate crystal. The question then becomes if the template matching approach for orientation analysis can be further optimized in regards to accuracy of the found final result. The new fast functionality allows for the systematic study of acquisition parameters such as camera length and processing steps such as background subtraction. Both simulated and experimental data sets, the later based on scanning precession electron diffraction of face-centered cubic materials, are utilized for these parameter tests. It is found that the camera length should be as small as experimentally reasonable to maximize to amount of diffraction spots. Further, logarithmic intensity scaling should be used with background subtraction in the form of difference of gaussians. It is also observed that the accuracy of the results is uneven across orientation space. In addition, tilt series are used as input. These prove useful, as in an experimental data set, as the found crystal orientation can be compared to the known experimental tilt step, a reference area that can be template matched more robustly or a known orientation relationship, as is the case with identifiable twin boundaries. Instead of comparing the cross-correlation score between experimental and simulated template, the misorientation angle between the template matched results and the expected value can be used as matching quality parameter. In addition, misindexations can be identified and the accuracy of the method as a whole quantified. Open source based template matching has allowed for speeding up and improving template matching and will lead to the technique being more utilized and its results being fairly valued.

Foreword

This is the master thesis for a degree in Physics at NTNU. The work was done between January 2021 and June 2022. All code development and data analysis is done by me unless otherwise explicitly mentioned. The experimental data used here was taken by Dipanwita Chatterjee and Emil Christians at the TEM Gemini Centre in Trondheim. I would like to thank my supervisor Ton for the excellent guidance and the many wonderful discussions we have had on all things SPED, template matching, science in general and life. I also thank Håkon for his help and suggestions, especially those involving code, python and orientation analysis. I thank Emil, Tina and everyone in the SPED user group at the TEM Gemini Centre for discussions and sharing ideas. Lastly, I thank my parents whom supported my decision to move all the way from Louisiana to Norway in the middle of a pandemic to study physics.

Contents

1 Introduction **1**

2 Theory **2**

2.1 Crystallography . . . 2

2.1.1 Crystal Structures . . . 2

2.1.2 Point Groups . . . 3

2.1.3 Space Groups . . . 3

2.1.4 Labeling Planes and Directions . . . 4

2.1.5 Representation of Crystal Orientations . . . 4

2.2 Diffraction . . . 6

2.2.1 Scattering . . . 6

2.2.2 Diffraction . . . 6

2.3 Electron Microscopy . . . 9

2.3.1 Hardware Principles . . . 9

2.3.2 Relevant Modes of TEM . . . 11

2.4 Data Processing . . . 12

2.4.1 Preprocessing . . . 12

2.4.2 Normalized Cross Correlation . . . 13

2.4.3 Diffraction Simulations . . . 14

2.4.4 Results and Visualisation . . . 14

3 Methods **15**

3.1 Materials and Preparation . . . 15

3.2 Microscope and Data Collection . . . 15

3.3 Hardware and Software . . . 15

3.4 Work Flow . . . 16

3.4.1 Evaluation based on Simulated Data . . . 16

3.4.2 Evaluating preprocessing using experimental data . . . 16

3.4.3 Evaluating template matching using "known features" in experimental data 17

4 Results **19**

4.1 Analysis based on simulated data . . . 19

4.1.1 Camera Length . . . 19

4.1.2 Intensity Re-scaling . . . 20

4.1.3 Orientation Library Step Size . . . 24
4.2 Preprocessing of Experimental Data . . . 26
4.2.1 Alignment . . . 26
4.2.2 Background Subtraction . . . 26
4.2.3 Effect of Calibration . . . 28
4.3 Examining Tilt Series . . . 29
4.3.1 Simulated Tilt Series . . . 30
4.3.2 Au Nanoparticles Tilt Series . . . 30
4.3.3 Cu and Si CPU Tilt Series . . . 31
4.3.4 Au Thin Film Tilt Series . . . 32

5 Discussion **35**
5.1 Analysis based on simulated data . . . 35
5.2 Preprocessing of Experimental Data . . . 37
5.3 Examining Tilt Series . . . 38
5.4 Optimized Setup . . . 40

6 Conclusion **43**

7 Future Work **44**

References **45**

A Code Contributions to Pyxem **47**
A.1 results_dict_to_crystal_map . . . 47
A.2 calibration_utils.py . . . 48

B Crystallographic Information Files used **53**
B.1 Si . . . 53
B.2 Au . . . 53
B.3 Cu . . . 54

C General Template Matching Notebook **55**

D Simulated data code **60**

Acronyms

bcc Body Centered Cubic. 3, 8

DP Diffraction Pattern. 16

fcc Face Centered Cubic. 2–5, 8, 15, 19, 44

FIB Focused Ion Beam. 15

IPF Inverse Pole Figure. 4, 14, 32

NCC Normalized Cross Correlation. 13, 14, 17, 19–21, 26–29, 35–37, 40, 41

PED Precession Electron Diffraction. 9

sc Simple Cubic. 2, 3

SPED Scanning Precession Electron Diffraction. 1, 11, 12, 15, 36

TEM Transmission Electron Microscope. 9, 15

1 Introduction

The properties of materials are dependent on their structure. The study and tailoring of these material properties is important whether it being the study of Al alloys for use in the manufacturing industries or novel materials to solve challenges the world is facing. The ordering of atoms in materials and especially the change in this order affects the properties as properties are, in general, direction dependent. To map and analyse the orientation of these materials is often an important part of the characterization of crystalline materials. Electron microscopy offers high spatial resolution for structural crystalline material analysis, allowing for the probing of the atomic structures of materials. This allowing for an impressive amount of information to be acquired. 4DSTEM in particular has become a hot topic to the fields of material physics and material sciences as it is flexible, powerful and allows for (semi-) automatic analysis with good resolution relative to large data sizes [21]. One form of 4DSTEM is Scanning Precession Electron Diffraction (SPED), [17], [24], where the electron beam is precessed. This leads to more stable patterns with more reflections, which aids in data processing to obtain crystal phase and orientation maps with nm-scale spatial resolution.

The 4D data stacks obtained from SPED can be analyzed by creating images based on selecting certain parts of the the 2D signal (virtual imaging), machine learning approaches to extract representative patterns [16] or comparison to a library of simulated templates from candidate materials at specified orientations (template matching) [23]. Template matching has become the default for phase and orientation analysis of SPED data. Early template matching developed by Edgar Rauch [23] and the basis for the most used commercial package, does not take advantage of relatively new computing technologies such as GPU integration or CPU multi-threading. The old standard template matching package, ASTAR [19], is a proprietary and closed source program that has the fundamental problem of being a 'black-box'. Other alternatives include Py4DSTEM [4], which is an open source implementation in python. Recent developments to Pyxem [12], in the form of *Fast Template Matching* from Niels Cautaerts et. all [5] has become available in the open source python environment. This form of template matching is indeed fast, allowing for CPU multi-threading, GPU integration with Nvidia processors, parallelization and clever manipulation of templates in orientation space to produce overall lower computation times than the alternatives. This, in turn, allows for the analysis of larger volumes of data or the use of higher quality input. It also allows a systematic evaluation of template matching method parameters such as image processing approaches applied to individual patterns and parameters used in simulation such as calibration.

The goal of this work is to test and utilize this recent implementation of template matching [5], to show its potential and find optimized process parameters to achieve better precision and accuracy from a more objective point of view, as Cautaerts et. all only test it in relation to ASTAR. Further more, this work will find and describe the challenges present in the template matching routine and attempt to find ways to work around them. For this work face-centered cubic is taken as example structure, as this highly symmetric structure is one of the most important conceptually and in engineering materials. As test data for this evaluation study simulated and experimental data sets from different sample geometries are used: nanoparticles, thin films and multiphase devices. Especially important, is the use of simulated input data as it allows analyzing the absolute accuracy of the template matching results versus a ground truth and thereby how the accuracy can be optimized by changing given parameters.

The work is structured as follows: first in Ch.2, relevant background theory on crystallography, diffraction, electron diffraction using a TEM, template matching work flow and relevant data processing steps will be presented. Details of methodology on the different samples as well as computational setup and how the template matching will be evaluated are in Ch.3. The results of different analysis and parameter studies based on both simulated data and experimental data are presented in Ch.4 and these results, observations and their impact on the efficacy of this template matching function are discussed in Ch.5. Finally, concrete suggestions based on these results will be given to improve the SPED based orientation mapping method. The used and constructed open-source code is given in the appendix to allow transparency and a basis for further developments for template based orientation mapping.

2 Theory

This chapter briefly covers relevant theories of crystal structures ([15], [13], [7]), electron diffraction ([29]), electron microscopy ([17], [27]) and data processing required for template matching ([5], [21], [23]).

2.1 Crystallography

2.1.1 Crystal Structures

In order to begin to describe the properties of different solid, it is necessary to first describe how the atoms in a crystal are arranged. A crystal is made of two parts, a basic structure or **unit cell** and a **lattice** that this unit cell periodically repeats itself within. The positions of atoms can be defined within a **Bravais lattice** in 3D as follows:

$$\vec{R} = n_1\vec{a} + n_2\vec{b} + n_3\vec{c} \tag{1}$$

where n_1, n_2 and n_3 are integers and $\vec{a}, \vec{b}$ and $\vec{c}$ are three *primitive*, linearly independent vectors [15].

The simplest lattice, the Simple Cubic (sc) lattice has the primitive vectors:

$$\vec{a} = a(1\ 0\ 0), \vec{b} = a(0\ 1\ 0), \vec{c} = a(0\ 0\ 1) \tag{2}$$

where a is the space between atoms. There are seven distinct crystal systems; cubic being the simplest most symmetric, with all side lengths and angles being equal and triclinic the least symmetric, with no side lengths or angles being equal. These systems can also categorized by their six lattice parameters: edge lengths a, b, c and angles between them α, β, γ. For the cubic lattices these are $a = b = c$ and $\alpha = \beta = \gamma = 90°$, while for example the hexagonal lattices' parameters are $a = b$ and $\alpha = \beta = 90°$ $\gamma = 120°$. Additionally, these systems can have different lattices and there are in total 14 Bravais lattices. The cubic system can for example be primitive(P), body-centered(I) or face-centered(F).

However simple cubic crystals are quite uncommon to find among monoatomic structures. A more common lattice found in nature, and consequently the lattice that both gold (Au) and copper (Cu), which will be the main crystalline materials used in this thesis, prefer Face Centered Cubic (fcc)[1]. The fcc lattice has atoms in both the corners and centers of the faces of a cube as in 2.1c. The primitive vectors, for an fcc lattice are given by:

$$\vec{a} = \frac{a}{2}(1\ 1\ 0),\ \vec{b} = \frac{a}{2}(1\ 0\ 1),\ \vec{c} = \frac{a}{2}(0\ 1\ 1) \tag{3}$$

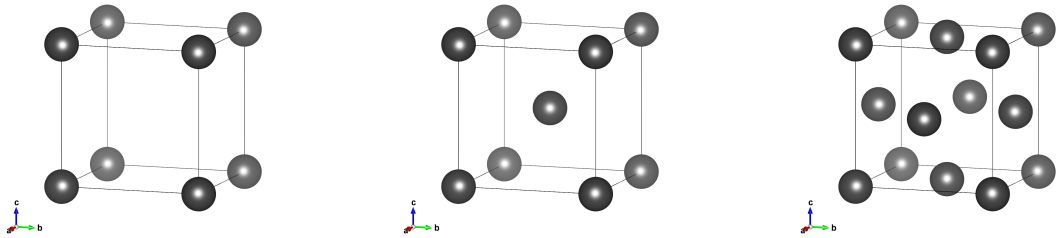
where a is the length between adjacent corners of the cube [15].

An important lattice for this work is the diamond structure, another cubic lattice and related to fcc and the structure that silicon (Si) takes; see 2.2a. It is formed by taking two fcc lattices and shifting one by $\frac{1}{4}$ along the body diagonal. It's primitive vectors are given by:

$$\vec{a} = \frac{a}{2}\hat{x} + \frac{a}{2}\hat{y},\ \vec{b} = \frac{a}{2}\hat{x} + \frac{a}{2}\hat{z},\ \vec{c} = \frac{a}{2}\hat{y} + \frac{a}{2}\hat{z} \tag{4}$$

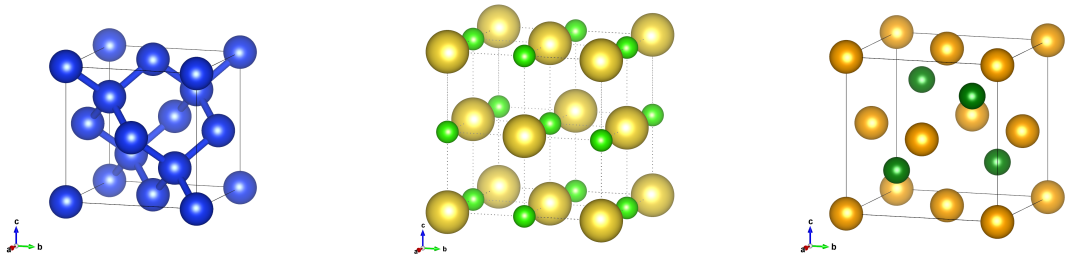
It is of course possible to to have compound structures; different elements forming the group at each lattice point. A composite structure that can be described using the introduced fcc lattice

[1] face centered cubic is also known as cubic close-packed and represents the most efficient packing mode of spheres in a cube.



(a) (b) (c)

Figure 2.1: Three cubic Bravais Lattices: (a) Simple Cubic (sc), (b) Body Centered Cubic (bcc), (c) Face Centered Cubic (fcc)



(a) (b) (c)

Figure 2.2: Examples of (a) diamond(Si), (b) rocksalt(NaCl) and (c) zincblende(GaAs) structures

is the rocksalt structure, that which NaCl commonly takes. This is created by again taking two fcc lattices but this time shifting one by $\frac{1}{2}$ along any lattice vector as they are all symmetric, see Figure 2.2b. By taking the diamond structure but alternating the atoms, the zincblende structure can be arrived at as in Figure 2.2c. These quite simple cubic systems with 1 or 2 elements can be easily described visually; however for more complex regular structures, a more systematically complete description is necessary.

2.1.2 Point Groups

A crystal can also be described by its symmetry. There are only a few symmetry elements required: rotation *1, 2, 3, 4, 6*, reflection m, inversion i or *-1* and rotation inversion. Combining these there are only 32 unique combinations, the 32 crystallographic point groups ranging from least symmetry: 1 in the Hermann–Mauguin notation to most symmetry. For monoatomic fcc the notation of point group is $\frac{4}{m}\bar{3}\frac{2}{m}$. The notation describes the symmetry elements of the structure for example in $\frac{4}{m}\bar{3}\frac{2}{m}$ the first, $\frac{4}{m}$, describes four fold, 90°, mirrors in the $\vec{a}, \vec{b}$ and $\vec{c}$ directions, the second, $\bar{3}$, describes a three fold around the central diagonals, and the third ,$\frac{2}{m}$, describes a two fold around the face diagonals. These Hermann–Mauguin notations are also often shortened, for example $m\bar{3}m$ is short for $\frac{4}{m}\bar{3}\frac{2}{m}$. The above introduced diamond and rock salt lattice have the same point group. The zincblende, although related via the deduction above, has the $\bar{4}3m$ point group. Point groups are relevant for describing and understanding properties such as conductivity and optical properties.

2.1.3 Space Groups

For this, the symmetry of the group at a lattice point(section 2.1.2) is combined with the symmetry of the lattice: the 14 Bravais lattices(section 2.1.1). There can be additional translation symmetries, a combination of a symmetry element as described above and a partial translation:

a reflection plane can become a glide plane (along some axis: a,b,c,g,d or n) and a rotation axis can become a screw axis. With proper book keeping, as not all combinations are valid, 230 crystallographic space groups can be fully described from this. For example Cu and Au are fcc with a point group of $m\bar{3}m$ and so have a space group of $Fm\bar{3}m$ or space group 225 and Si has a space group of $Fd\bar{3}m$ or 227; compared to 225 a m is replaced by a diagonal glide plane: d.

The 230 space groups are listed in the International Table of crystallography, describing all the symmetry and possible lattice points. Thereby crystals can be created (using the specific symmetry operations), knowing the space group, elements and their lattice positions and the six lattice parameters. This information is summarized in a crystal information file, .cif, used to create, by software as used in present work, models or do calculations on a given crystalline material. In this work .cif files from [11] are used for simulating the candidate materials and calculating diffraction patterns rather than defining them directed as described in 2.1.1 see appendix B for list.

2.1.4 Labeling Planes and Directions

In order to further describe crystals, indexing of directions and planes in crystals using **zone axes** and **Miller indices**. A zone axis, generally [U V W], expresses distance based on the lattice constants a_1, a_2 and a_3 as the whole numbers U, V and W. A group of symmetrically equivalent zones axes is generally $\langle$ U V W $\rangle$, so $\langle 1\ 0\ 0 \rangle$ will describe all six of the cubic primitive vectors, both positive and negative, including [1 0 0], [0 1 0] and [0 0 $\bar{1}$]. See figure 2.3 for visual example. Planes are described by Miller indices, generally (hkl) where hkl are, similar to the zone axes, whole numbers describing the plane intercepts of $\vec{a_1}$, $\vec{a_2}$ and $\vec{a_3}$. Miller indices can also be used to describe a family of symmetrical planes, generally {hkl}. For example {1 0 0} describes all six equivalent faces of a cube [13].

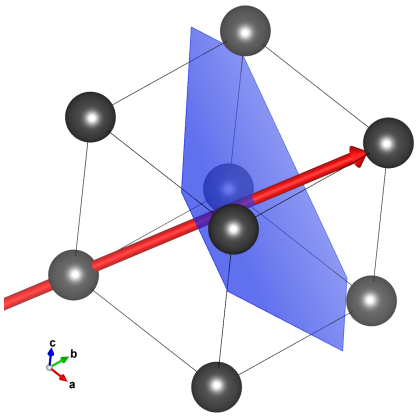
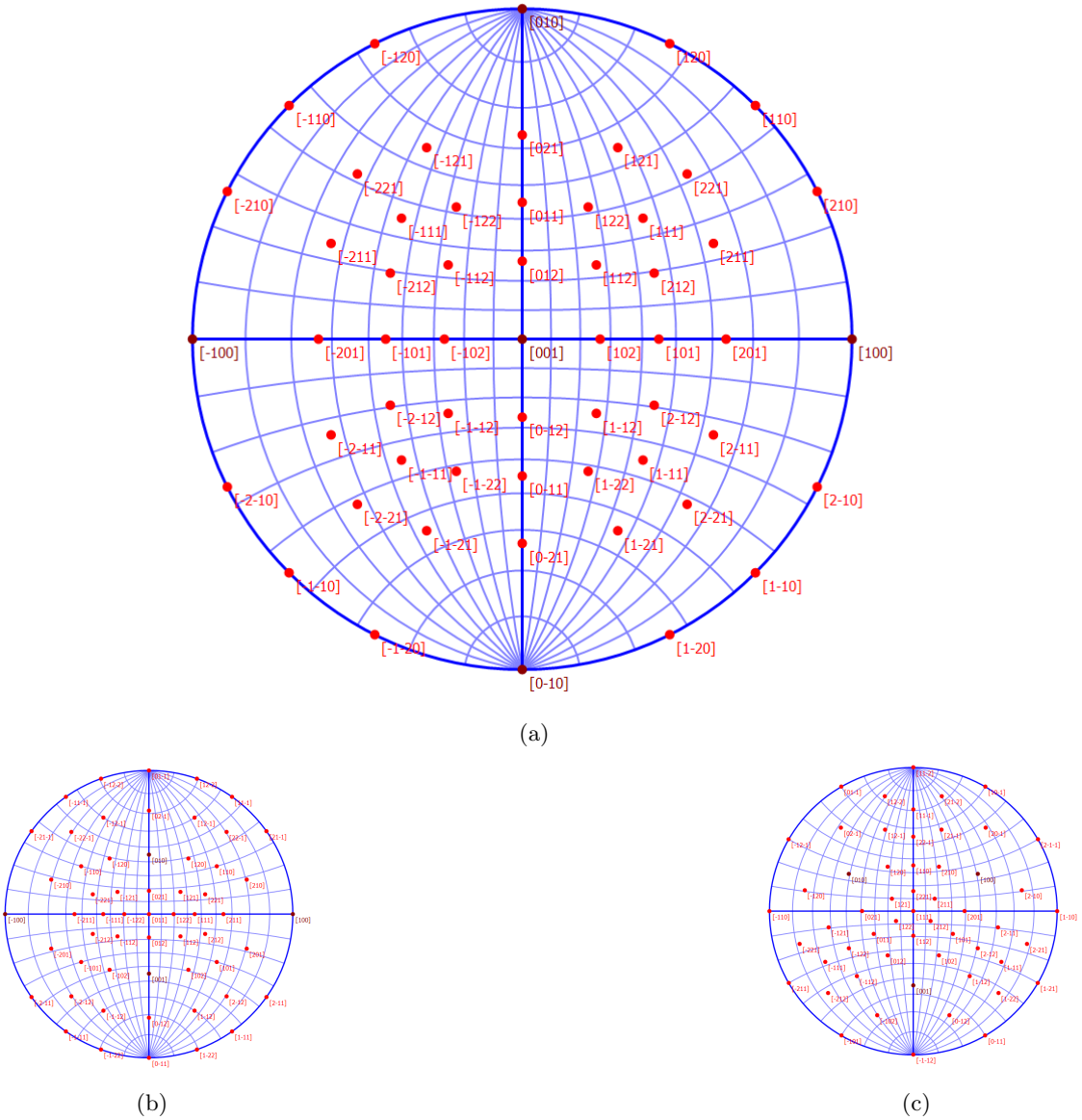


Figure 2.3: Lattice vector [111] and the plane, {111}, it is normal to

2.1.5 Representation of Crystal Orientations

Two common ways in which 3D orientations are visualized in 2D representations are with a **pole figure** and an **Inverse Pole Figure (IPF)**. A pole figure is created by placing a crystal cell in the center of a unit sphere, where the crystal's plane normals intersect this sphere are its **poles**. The pole figure is then at least two of these poles that come from non-parallel planes. Then by projecting these poles down to the equatorial plane, a **stereographic projection** can be created. The stereographic projection, in planes or directions, can be used to analyze relationships between different planes/directions, such as the angle between them. Some example stereographic poles are given in Fig. 2.4. An IPF is created by instead of rotating the crystal at the center of the sphere and taking its poles, the sphere is rotated around the crystal and its orientation is defined based on the coordinate system of the specimen [7]. The IPF is often reduced to a characteristic **reduced zone** based on the symmetry of the point group.

Instead of visualizing orientations, **euler angles** can be used to describe them in a more direct way. These euler angles are a set of three angles that describe a set of chained rotations of an object. For crystallography this is often done the Bunge convention, where the order of the rotations is $ZX'Z''$. These angles are normally given for on object with respect to the *sample coordinates*, the Z direction being along the beam direction and the X and Y are 90° from each other along the specimen. Instead of visualizing or describing a specific orientation of the crystal, it can be more relevant to describe an orientation with respect to another orientation; called a misorientation. Misorientations can be useful for discussing differences in orientations between grains or between tilts and will be further discussed in sections 4 and 5.



(a)

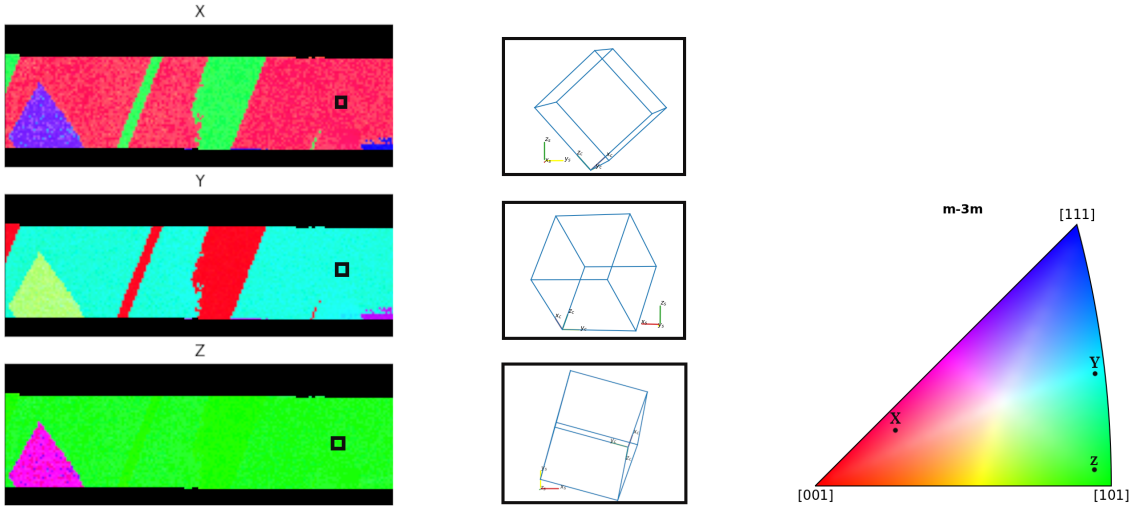
(b)

(c)

Figure 2.4: Example of stereographic projections of an fcc structure along the (a) [001], (b) [011], (c) [111] poles.

When mapping orientations it is often useful to color an image of the sample in real space with colors based on orientation. This is done by assigning colors across the reduced zone in a direction and coloring a real space image based on the color corresponding to orientation. An example is given in Fig. 2.5 where the coloring for the three orthogonal directions, the real space orientation of unit cell and the IPF depicting the color coding are shown [20]. It is important to note that one orientation map will only give the orientation in one direction, i.e. along the z direction and as

such to get the full orientation map at least two directions must be used. For this work z is fixed to the direction of the electron beam .



(a) (b) (c)

Figure 2.5: (a) Colored orientation map of sample C with (b) unit cell orientations and (c) colored IPF showing the orientation to color relation.

2.2 Diffraction

2.2.1 Scattering

Electrons are charged and have low mass, which allows for them to be easily scattered when passing near atoms, either by the nucleus or by the electrons of the atoms. Electron scattering is normally split by two criteria: **elastic** vs **inelastic** and **coherent** vs **incoherent**. If there is no energy loss then the scattering is elastic, if there is energy loss it is inelastic. The coherence of the scattered electrons deals with their wave nature. Two electrons are coherent if they have the same wavelength and phase. Finally, scattering can be direction dependent. It is differentiated by its angle of scattering, it is **forward scattered** if it is $< 90°$ or **backscattered** if it is $> 90°$. Generally speaking, elastic scattering is coherent for thin crystalline samples forward scatterd and has a low scattering angle(1-10°).

2.2.2 Diffraction

Coherent elastic scatter of a wave with wave length λ similar to the interatomic lattice plane spacing d: θ_B (for Bragg) is the angle between the wave vector and the atomic plane and the scattering angle is $2\theta_B$ [29]. This is Braggs' law, written:

$$2d_{hkl} \sin \theta_B = n\lambda \tag{5}$$

In 2.1 the crystal is described in real space, however it can also be described in so called reciprocal space. Rewriting Braggs' law as:

$$\frac{2 \sin \theta_B}{\lambda} = \frac{n}{d} = |\vec{K}| \tag{6}$$

This new vector $\vec{K}$ is reciprocally related to d. A reciprocal-lattice vector can be defined similar to 1 as:

$$\vec{R^*} = m_1\vec{a^*} + m_2\vec{b^*} + m_3\vec{c^*} \tag{7}$$

These translation vectors $\vec{a^*}$,$\vec{b^*}$ and $\vec{c^*}$ are defined by:

$$\vec{a^*} \cdot \vec{b} = \vec{a^*} \cdot \vec{c} = \vec{b^*} \cdot \vec{c} = \vec{b^*} \cdot \vec{a} = \vec{c^*} \cdot \vec{b} = \vec{c^*} \cdot \vec{a} = 0 \tag{8}$$

and

$$\vec{a^*} \cdot \vec{a} = 1; \vec{b^*} \cdot \vec{b} = 1; \vec{c^*} \cdot \vec{c} = 1 \tag{9}$$

From this a reciprocal vector $\vec{g_{hkl}}$ can be defined as:

$$\vec{g_{hkl}} = h\vec{a^*} + k\vec{b^*} + l\vec{c^*} \tag{10}$$

where h, k, l define the plane (hkl) and the length of $\vec{g_{hkl}}$ is defined as:

$$|\vec{g_{hkl}}| = \frac{1}{d_{hkl}} \tag{11}$$

Putting all of this together, the conditions for Bragg diffraction can be defined as:

$$\vec{K} \cdot \vec{r_n} = N \tag{12}$$

or even more simply:

$$Q = G \tag{13}$$

which are the Laue conditions for different directions. The condition is slightly relaxed as $\vec{K}$ can also be defined as $\vec{g_{hkl}} + s$ with s being the **excitation error** of small deviations from the absolute Bragg condition that will still produce diffraction spots. Convoluting the reciprocal lattice with a reciprocal-lattice rod(relrod), that has a length of s, centered on each lattice point gives a 3D grid of relrods that is used to describe the points that produce diffraction spots. By drawing a sphere of radius $\frac{2\pi}{\lambda}$ onto the lattice, and where this **Ewald sphere** intersects the relrods in the lattice diffraction spots are produced. This is a practical visual representation of when diffraction occurs and schematically shown in Fig. 2.6.

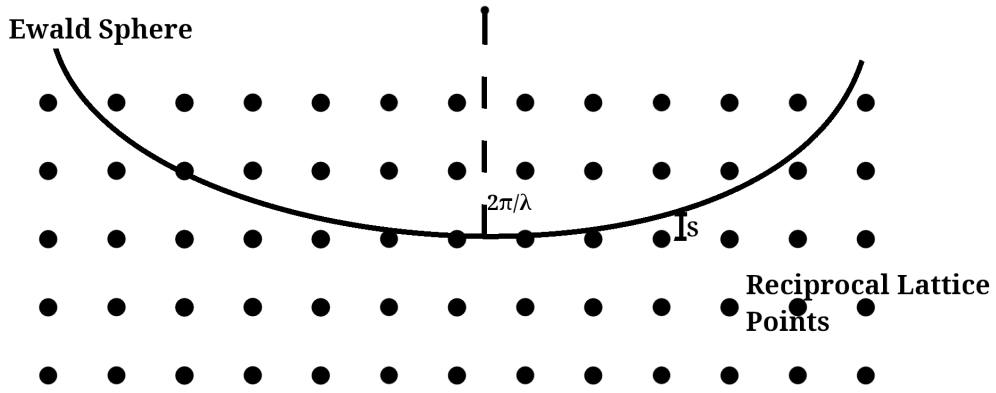
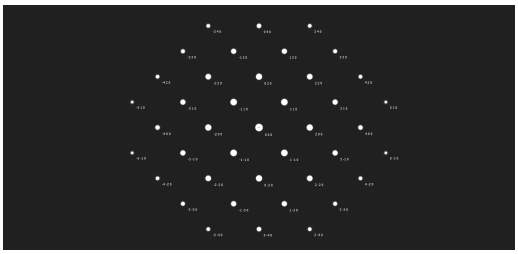


Figure 2.6: Ewald sphere over reciprocal lattice

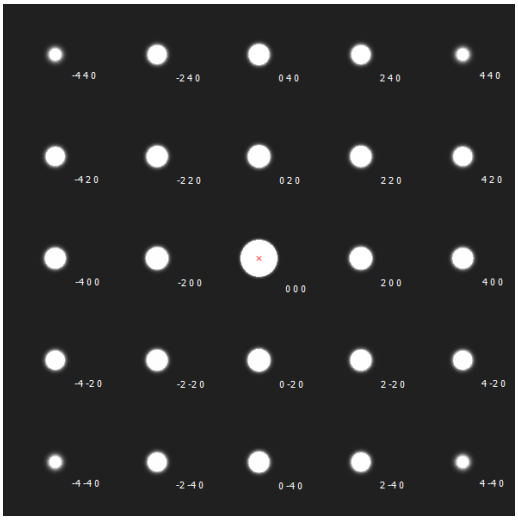
When $\vec{K} = \vec{g_{hkl}}$ is satisfied the intensity of the coherent diffraction spot is defined as

$$I(q) = |F_G|^2 \tag{14}$$

$$NF_G = N \int_{cell} dV n(\vec{r}) \exp -i\vec{g} \cdot \vec{r} \tag{15}$$



(a)



(b)

Figure 2.7: Example diffraction patterns of (a) bcc and (b) fcc both at the [001]

This $\vec{F_G}$ is the **structure factor**. The structure factor of the basis is defined as

$$F_G = \sum f_j \exp -i2\pi(hx_j + ky_j + lz_j) \tag{16}$$

For a primative lattice, one basis per unit cell, all planes will have a non-zero F_G Using the bcc lattice as an example; the basis has identical atoms at (0 0 0) and $(\frac{1}{2}\frac{1}{2}\frac{1}{2})$ so the structure factor for bcc is:

$$F(hkl) = f(1 + \exp -i\pi(h + k + l)) \tag{17}$$

This creates the **kinematic diffraction conditions** for bcc as:

$$\begin{aligned} F = 2\text{f for } h + k + l = \text{even integer} \\ F = 0 \text{ for } h + k + l = \text{odd integer} \end{aligned} \tag{18}$$

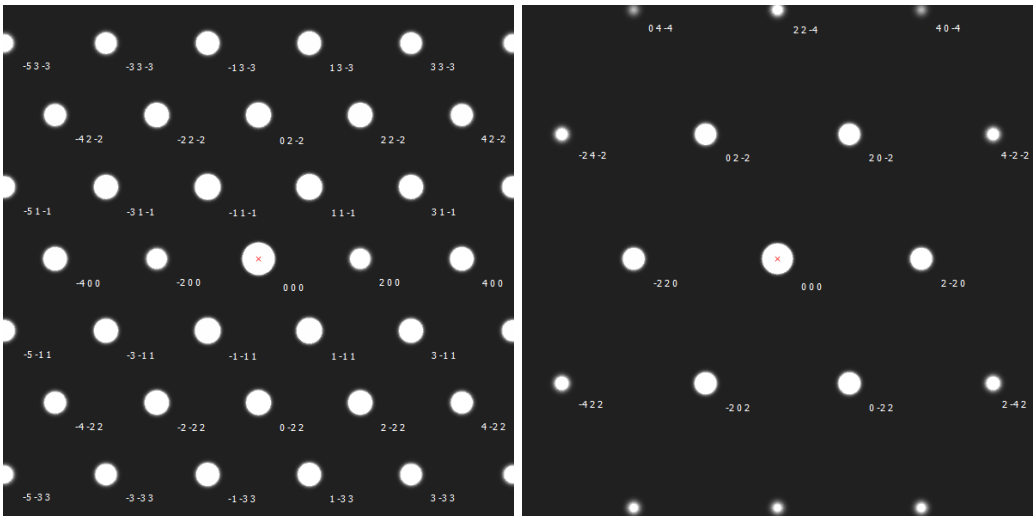
Thus for bcc structures, the diffraction pattern at (100) or (111) will have no intensity, due to destructive interference from that plane, but will contain spots at (200) and (110). Refer to a Fig. 2.7 for the [001] zone of bcc.

The same can be done for fcc giving:

$$\begin{aligned} F = 4\text{f for } h, k, l = \text{all odd or all even} \\ F = 0 \text{ for } h, k, l = \text{mixed odd and even} \end{aligned} \tag{19}$$

These diffraction patterns are dependent on the relation between the crystal orientation and the incoming electrons. Different orientations of the crystal will create distinct diffraction patterns like those in Fig. 2.7. For a polyatomic structures, the structure factor is no dependant on the relation between the to different atoms, for example rocksalt has diffraction conditions:

$$\begin{aligned} F = (4(f_{Na} - f_{Cl}))^2 \text{ for } h, k, l = \text{all odd} \\ F = (4(f_{Na} + f_{Cl}))^2 \text{ for } h, k, l = \text{all even} \\ F = 0 \text{ for } h, k, l = \text{mixed odd and even} \end{aligned} \tag{20}$$



(a) (b)

Figure 2.8: Example diffraction patterns of fcc at (a) [011] (b) [111]

An important note is that these rules and the intensity equation are only true for purely kinematic scattering, in an experimental setting the diffraction becomes more complex as electrons can interact with other already diffracted electrons inside the specimen. These *dynamical effects* can extinguish spots that are allowed or create spots that are forbidden, which can sometimes lead to unexpected results. For example a diamond structure has conditions:

$$\begin{aligned} F &= 4\text{f for } h,k,l = \text{all odd} \\ F &= 4\text{f for } h,k,l = \text{all even and } h+k+l = 4n \\ F &= 0 \text{ for } h,k,l = \text{mixed odd and even} \end{aligned} \tag{21}$$

Given these conditions, (200) would be forbidden, however a sample could exhibit (200) spots that come from the (111) spots interfering constructively, shown in Fig. 2.9.

2.3 Electron Microscopy

2.3.1 Hardware Principles

A Transmission Electron Microscope (TEM) can be split into five principle stages; an electron source, illumination stage with condenser lenses, objective stage with the objective lens and sample, magnification stage with intermediate lenses and projector lens and finally observation and/or recording device. The electron source is simply what creates the electrons at a given energy, an example being a field-emission gun. The illumination part contains a set of condenser lenses that are used to focus and direct the electron beam onto the sample. The objective lens is the primary lens for forming an image from the sample that sits inside this lens. The magnification part consists of multiple intermediate lenses that are used to adjust the magnification and to shift between real space and reciprocal space. These lenses are electromagnetic lenses that can change the trajectory of the electron beam. In each segment there are also apertures to limit the beam or select a certain scattered beam and stigmators to obtain a round beam or isotropic signal. This is only the principal components, there are of course additional components especially for specific modes of TEM. For example, when using Precession Electron Diffraction (PED) there are deflectors to tilt the beam about the optic axis and deflectors to correct the image for the rotating beam, explained further in 2.3.2.

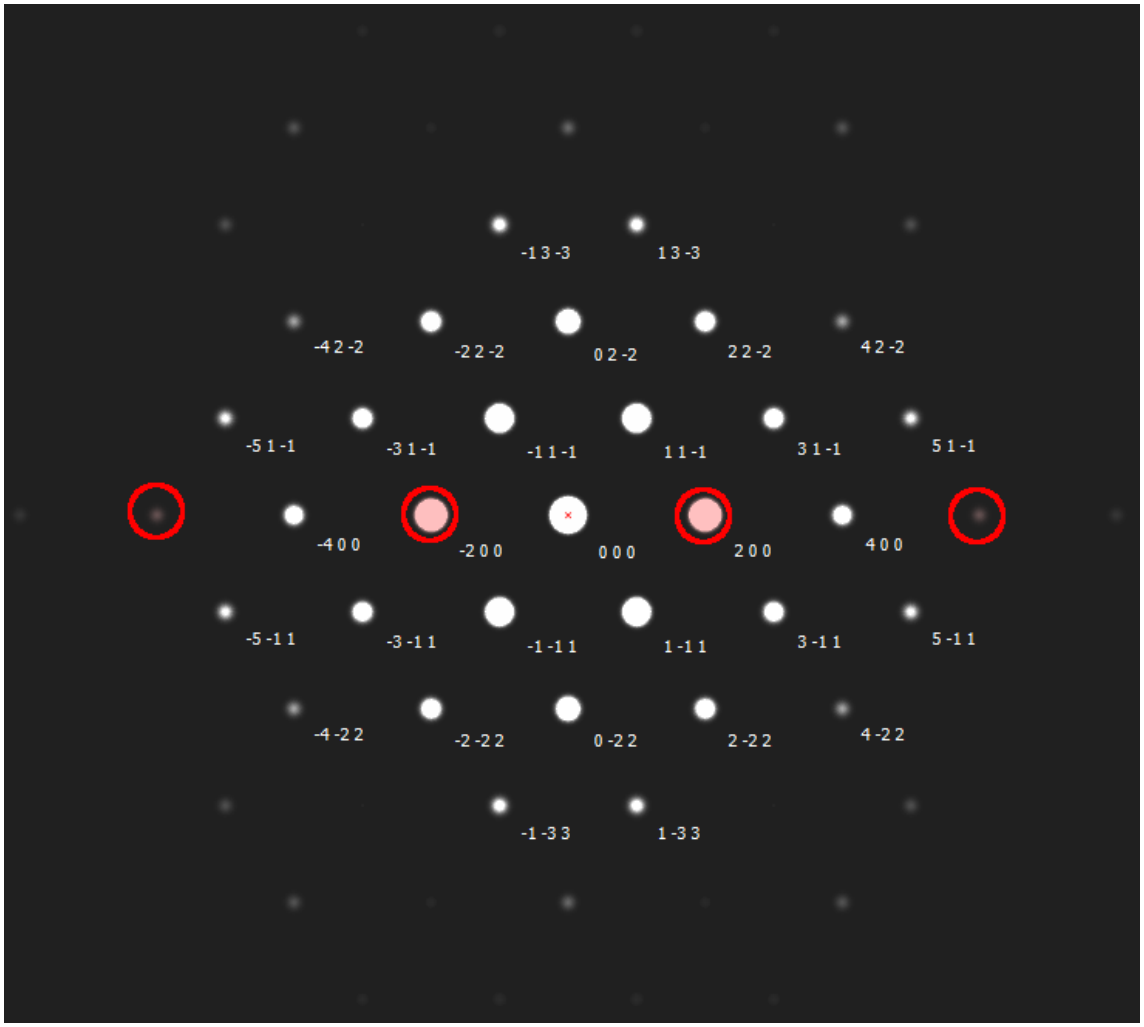


Figure 2.9: Simulated diffraction pattern of Si(diamond cubic) with kinematically forbidden spots in red

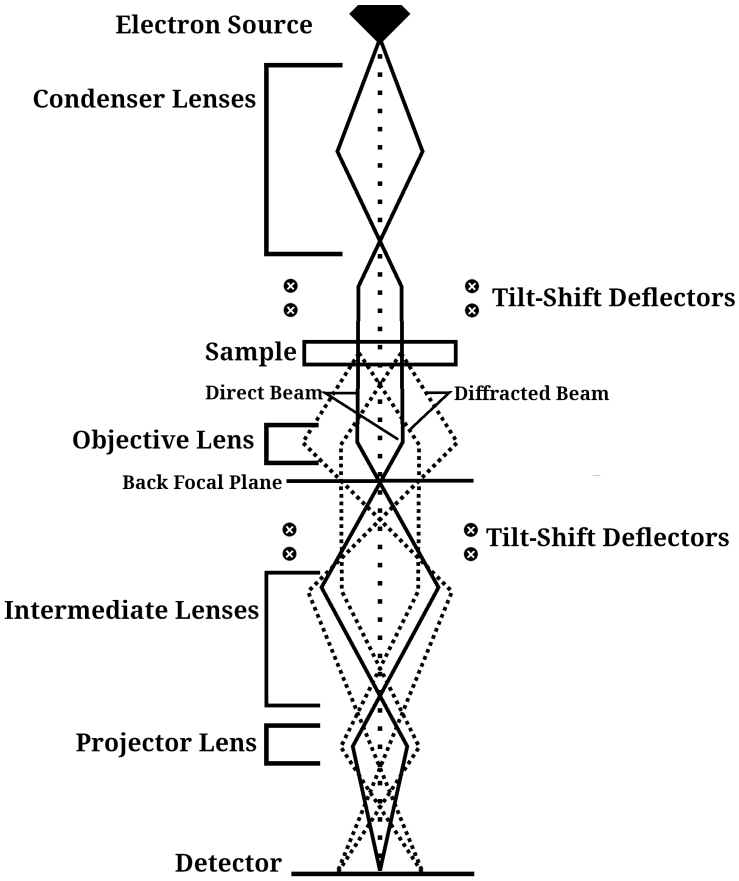


Figure 2.10: Schematic of TEM in diffraction mode

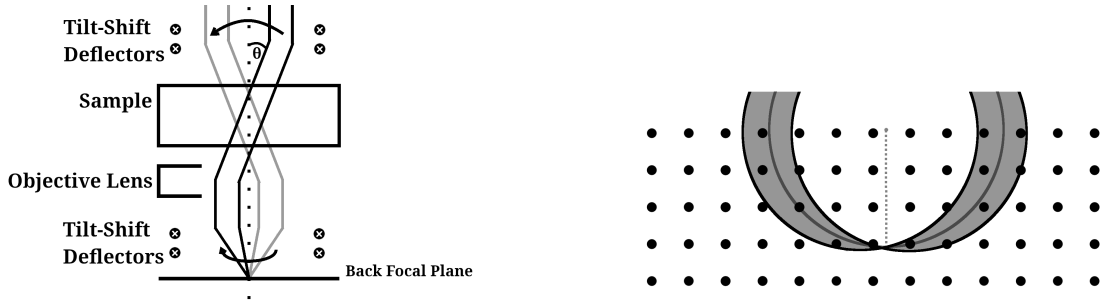
2.3.2 Relevant Modes of TEM

While there are a great many possible modes that a TEM can be used for utilizing the different scattering phenomena described in 2.2.2. Techniques include dark field imaging, high-resolution TEM, electron energy loss spectroscopy(EELS) and energy-dispersive X-ray spectroscopy(EDS); the relevant modes for this work are Scanning Transmission Electron Microscopy(STEM), Precession Electron Diffraction(PED) and Scanning Precession Electron Diffraction (SPED). The later two are central in the present work and dedicated imaging and spectroscopy techniques are not further addressed here.

Scanning Electron Diffraction Instead of having a relatively large parallel beam hitting the sample and using a selected area aperture in an image plane under the specimen; a convergent, more focused electron beam can be used to hit more localized parts of the sample: a few nm instead of ¿100 nm as in selected area diffraction. This beam will produce diffracted spots more as discs than the more point-like spots from a parallel beam. By rastering or scanning the beam across the sample and recording an image at a specified frequency, a set of diffraction patterns can be recorded with each one representing a small(normally around 1-2 nm) part of real space.

Precession Electron Diffraction The electron beam can be precessed about the optic axis at some constant precession angle with a specific frequency. As the beam rotates, multiple diffraction patterns are captured, which can then be integrated to a create a single 'mean' pattern. It is important to note that after the beam has passed through the sample, it is rotated back(descanned) in order to correct the image since the beam is being rotated. The resultant pattern has more reflections as more of reciprocal space is scanned (shaded region in Fig. 2.11(b)). In addition, the rotational summed pattern, due to the averaging of many slightly different angles,

will appear more kinematic like [27].



(a) (b)

Figure 2.11: (a)Schematic view of a precessed beam through a sample and (b) Precessed Ewald sphere over a reciprocal lattice

Scanning Precession Electron Diffraction By scanning the PED beam across a specimen, many different precessed diffraction patterns can by acquired for the sample creating a Scanning Precession Electron Diffraction (SPED) data set. This SPED data set is a 4D (two real space, two reciprocal space; written as $(x, y|s_x, s_y)$) set where each pixel in real space has a PED pattern. This allows for collection of nanometre scale orientation and phase information from a sample that can then be used for texture, strain or grain size analysis [17].

2.4 Data Processing

Once the data set, a 4D data stack, from SPED has been collected, it is processed based on the procedure in figure 2.12. This procedure is called template matching where the experimental patterns at each scanned point are compared to the calculated patterns in different directions for candidate phase(s) [23], [5], [21]. This will give phase and orientation maps.

The SPED data can be processed differently, for example virtual imaging, a selected part of signal space is represented in the scanning space, or machine learning approaches selecting representative patterns [16]. The *Data* and *Crystal Structures* sections have been discussed above, here the remaining steps will be explained.

2.4.1 Preprocessing

Diffraction Pattern Stack Alignment Alignment is an essential step for the data processing of SPED data. Here the individual diffraction patterns are aligned so that the center of the direct beam is in the center of all of the images. This is necessary as there can be drift of the patterns over the dataset. There are three common methods used in Pyxem [12] used for finding the shift required for aligning: cross correlation, blurring and interpolation.

- **Cross correlate**: The shift is calculated relative to an circle perimeter. The circle can be refined across a range of radii during the centring procedure to improve performance in regions where the direct beam size changes, e.g. during sample thickness variation.
- **Blur**: Estimates direct beam position by blurring the image with a large Gaussian kernel and finding the maximum.
- **Interpolate**: Finds the center of the primary beam in the image by summing along X/Y directions and finding the position along the two directions independently.

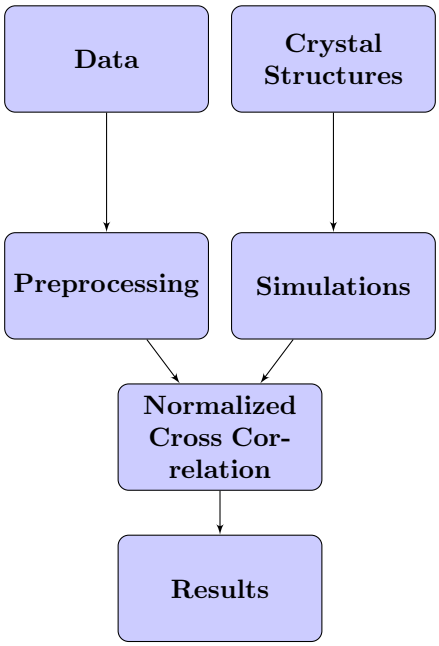


Figure 2.12: Standard Data Workflow: Schematic representation of analyzing template matching using the normalized cross correlation scores.

Background Noise Subtraction Another important step, particularly for template matching, is background subtraction. The diffraction patterns will have a background noise that most consists of a gaussian profile centered on the direct beam from the substrate the sample is prepared on. The methods used for this are common methods used for other types of signal noise reduction, difference of gaussians, median kernel, radial median kernel and h-dome are used in Pyxem [12].

- **Difference of Gaussians**: A edge enhancement filter that works by subtracting the original image from a gaussian blurred image of given gaussian size.
- **Median Kernel**: A smoothing filter that keeps edges by replacing each entry with the median of neighboring, or more, entries.
- **Radial Median**: Similar to median kernel but instead of neighbors or grouped neighbors, it uses a median based off of radial distance for each pixel.
- **H-Dome**: A method that finds local maxima and then removes those that are lower than a given parameter.

The results for these different background subtraction methods are discussed further in section 4.2.2.

2.4.2 Normalized Cross Correlation

The patterns and simulations are compared by calculating the Normalized Cross Correlation (NCC) score:

$$Q = \frac{\sum_{j=1}^{m} P(x_j, y_j)T(x_j, y_j)}{\sqrt{\sum_{j=1}^{m} P(x_j, y_j)^2}\sqrt{\sum_{j=1}^{m} T(x_j, y_j)^2}} \tag{22}$$

Then the template with the highest NCC score is taken as the 'best' match for a given pattern and it moves on to the next pattern. In essence this equation multiplies the intensity from the pixels within the pattern($P(x_j, y_j)$) and the template($T(x_j, y_j)$) and then normalizes the summed result. So for high intensity in both pattern and template on the same pixels gives a higher NCC

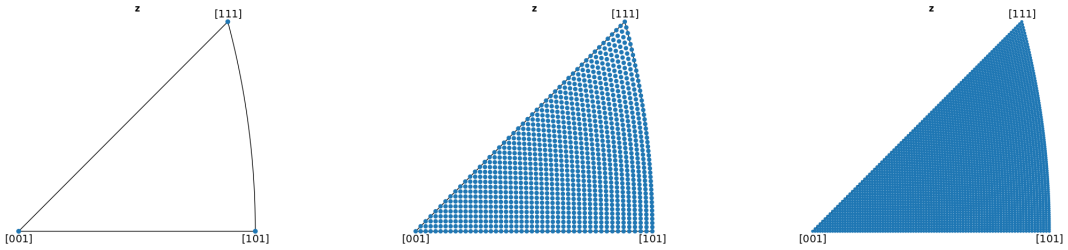
score, which equates to a 'better' match, as the highest overall score is chosen as the match. It is important to note that this is not preformed on every single pixel, but instead only the peaks of the diffraction spots in the simulations, to limit the computation required. NCC is used as it fast but still produces results similar to more computationally demanding methods [5].

2.4.3 Diffraction Simulations

An important step of template matching is simulating prospective patterns to which the experimental patterns are compared. The simulations follow the principles as described in section 2.2. In Diffsims this is done by first calculating the reciprocal lattice of the given structure and finding the points within the sphere $\frac{2}{\lambda}$. It then checks the Bragg condition, $\sin\theta = \frac{\lambda}{2d_{hkl}}$ for each reciprocal point. The intensity each of these kinematically allowed reflections is given by $I_{hkl} = F_{hkl}F^*_{hkl}$.

It is important to simulate the necessary orientations evenly and keeping in mind the symmetry of the system to limit the amount of computations necessary. For instance, in a cubic structure it is unnecessary to simulate every possible orientation because of the many different symmetries, so the simulations are only done for the reduced zone. Another simplification done in Pyxem, is to only simulate one pattern with zero inplane angle(the first euler angle) and to instead simply rotate that pattern over 360°; this is more computationally efficient than simulating all the patterns [5].

Gridding across orientation space is done by producing an array of beam directions, within the stereographic triangle of the relevant crystal system based on a mesh of the unit sphere. All of this together gives a computationally efficient setup for simulating the necessary diffraction spots used in template matching [5]. The resolution or the grid is given in degree and equates to the largest angle between any nearest template. For example, to grid over $m3m$ space at 45° resolution would give the three corner orientations of the IPF, at 1°: 1081 orientations are given and at 0.5°: 4186 evenly space orientations are returned.



(a) (b) (c)

Figure 2.13: Orientation grids of equal angular spacing in Z-IPF for resolutions of (a) 45°, (b) 1°, (c) 0.5°

2.4.4 Results and Visualisation

The results acquired from the template matching method consist of phase, orientation and NCC score. The phase returned as the match is simply whichever phase matches best, assuming there are multiple prospective phases in the library. Orientations are in the form of euler angles, as mentioned in 2.1.5. Euler angles are used instead of other orientation representations for computational reasons, an euler angle is only three numbers while for instance a rotation matrix is nine. It is also possible to record more than just the orientations with the highest NCC score, giving a ranked list of orientations, allowing for the potential for post-processing and quality metrics. Along with the orientations the NCC scores are also recorded, which can also be utilized in crystallographic analysis, which is discussed in in sections 4 and 5.

The visualisation of results is done principally in Orix [1], allowing for phase maps (simple maps of which phase matches best), orientation maps (as seen in figure 2.5), correlation maps and more.

3 Methods

3.1 Materials and Preparation

Three different samples are used in this work:

A: Au nanoparticles

B: Au thin film

C: CPU device

Sample A: The Au nanoparticles (fcc) were synthesized by a wet-chemical route as described by Singh et al. [25] by Dr S. Bandyopadhyay at the Department of Chemical process technology, NTNU. The specimen was prepared, by A. van Helvoort, by drop casting the sample further in a deionized water diluted suspension on a 300 mesh Cu grid with a holey C support film and drying the solution slowly in the air at room temperature. Well-distributed nanoparticles reduce the risk of overlap during tilt series.

Sample B: The thin Au films (fcc) were deposited by DC sputtering on 20 nm thick amorphous SiO_2 windows of commercially available TEM grids (TEMwindows.com). A thin metallic adhesion layer Ti was first deposited to avoid delamination of the Au film. The specimen was prepared by M. Heinig at DTU and further details can be found in [9]. The thickness of the amorphous Ti was approximately 1.0 nm and the polycrystalline Au film was 10 nm thick. This sample represent a polycrystalline film. The small thickness reduce risk over overlap and inelastic contributions. The sample contains a high density of $\Sigma 3$ twins

Sample C: A CPU chip [Still details from Magnus/Emil] were a TEM specimen was extracted by Focused Ion Beam (FIB) lift-out. The FIB used was a FEI Helios Dualbeam and done by M. Nord [check] using a 30 kV Ga beam and final polishing was done at 5 kV The specimen thickness was ca. 120 nm. This specimen represent device structures and contains a 111-Si substrate (diamond fcc) useful for calibrations, polycrystalline metallic channels (not studied in this work) and a thick poly-crystalline Cu film (fcc). TEM of this specimen has not been published so far.

3.2 Microscope and Data Collection

All data is collected on a JEOL JEM 2100F at 200 kV equipped with a Nanomegas ASTAR scanning precession system. The microscope was aligned following the principles of Barnard et al. [2]. A probe size of 1 nm, camera length of 12 or 20 cm, a precession angle of 0 (non-precessed reference data) or 1°, exposure time 1 ms (unprocessed) and 10 ms (precessed) per pixel and a precession frequency of 100 Hz. The data stacks were collected on a 256x256 Medipix direct electron detector using a QuantumDetectors Merlin system. The data of sample A and C were collected by Dr C. Christiansen and data of sample B was collected by Dr. D. Chatterjee. In Table 3.1 the data sets used in this study are presented.

3.3 Hardware and Software

Most of the template matching and data processing is done a PC with a NVIDIA GTX 1060 6GB possessing 1280 CUDA cores for the template matching, 16 GB of RAM and an Intel i5-7600K CPU. When necessary due to RAM or computational limitations, the IDUN cluster at NTNU was used. All the computational work was done in Python with the primary packages being Hyperspy[22] development version(now release 1.7) that provides the basis for how signals are stored, Pyxem[12] development version (now release 0.14.1) for most 4D SPED processing and template matching, Diffsims[6] development version (now release 0.5) for the diffraction simulations

Table 3.1: Data Sets Used in this Study: All datasets taken at a camera length of 12cm. All datasets have a pixel count of 16384, which gives a raw data size of 2GB each or 10GB for a set of five tilts.

	Au Nanoparticles(A)	Au Thin Film(B)	Cu and Si CPU(C)
Non-Precessed	zero tilt	zero tilt	zero tilt
Precessed (1°) Tilt Series	zero tilt zero tilt +5°x zero tilt +10°x - -	zero tilt zero tilt +5°x zero tilt +10°x zero tilt +5°y zero tilt +10°y	zero tilt zero tilt +5°x zero tilt +10°x zero tilt +5°y zero tilt +10°y
Size (Real|Reciprocal)	128x128|256x256	128x128|256x256	256x64|256x256
Step Size	3 nm	5 nm	11 nm

and Orix[1] development version (now release 0.9) for handling and plotting orientations. The used code, given in the form of Juypter Notebooks, is in appendix A. For experimental data sets, the following preprocessing is applied: data is first stored in a .mib file but is converted to a .hdf5 or .hspy for further use using Pxyem and Hyperspy. Once the data is in a usable format for Hyperspy and Pyxem, the data can be processed. The individual DPs are then centered as per 2.4.1 and any distortions corrected for with an affine transformation.

3.4 Work Flow

3.4.1 Evaluation based on Simulated Data

Simulated data is used to test the matching and preprocessing parameters of intensity scaling, simulation resolution and calibration in a controlled way. In place of the normal acquired data, the simulations used for the matching are first are given a guassian spread on the diffraction spots and then converted to hyperspy signal. No artificial noise is added to simulated data, the effect of noise will be evaluated using experimental data in the other parts. Here the following different parameters can be varied; calibration, camera length, intensity scaling, simulation library angular resolution(of 0.1°, 0.25°, 0.5°, 1°, 2°), and the crystal orientations themselves These results are then compared to the known "ground truth" orientations of the simulated DPs and the matched orientations. The simulated data is also used to create artificial tilt series to test how the matching is affected and can be used to evaluate experimental tilt series. The analysis in this part is depicted in Figure 3.1.

This approach allows for a ground truth to always be known so that parameters can be tested and their effects seen directly from resultant misorientations without the additional variables that experimental data comes with, such as artifacts or noise.

3.4.2 Evaluating preprocessing using experimental data

Experimental is also used to test preprocessing steps: intensity scaling, calibration and background subtraction. For the intensity scaling and calibration, this is done to reinforce the results from simulations. Testing the background subtraction is only done on experimental data. The data is given background subtraction as per 2.4 and any other major effects such as gaussian bluring or an intensity cutoff. The optimal calibration is found by taking prospective patterns and matching these many times while varying the calibration to find the highest correlation score. Once the experimental data is processed, it can undergo template matching. Here any intensity scaling such as logarithmic scale is done as well. Once the template matching is completed the results can be put into a Orix crystal map object that allows for easy saving, plotting, and further analysis.

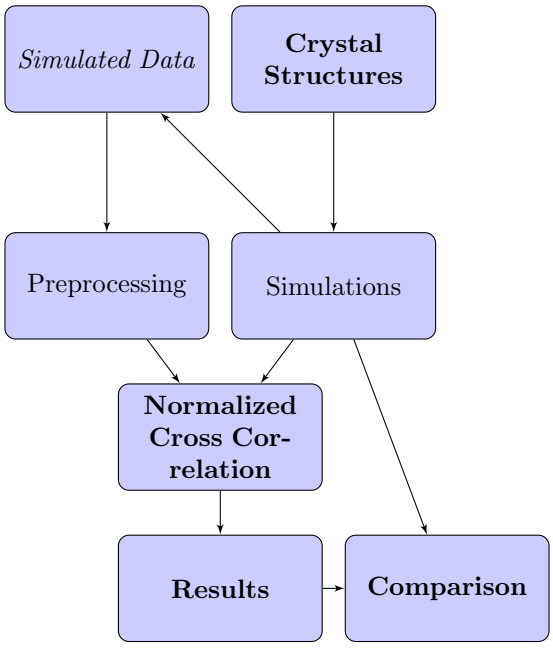


Figure 3.1: Simulated Data Workflow: Schematic representation of analyzing template matching using simulated input.

3.4.3 Evaluating template matching using "known features" in experimental data

For this part, experimental data is used as in 3.4.2. The updated work flow is seen in figure 3.2. However instead of optimizing for the NCC score, a known or knowns can be used to evaluated the template matching results. The difference between the tilts is used for all samples: A, B and C. In addition, sample B and C had a know orientation relation between some grains: Σ3 twins in Au and Cu respectively, which is a known 60° misorientation. Sample C has Si as a known feature at a known orientation(111) which allows for easy calibration and another way to track the tilts and compare with the Cu. Generally, there are not knowns present, like a tilt series or known orientation relation, only a library of candidate templates. Using the ranking correlation score can be used to analyze steps in the procedures as calibration and background handling. This will be done prior to analyzing tilt series with knowns.

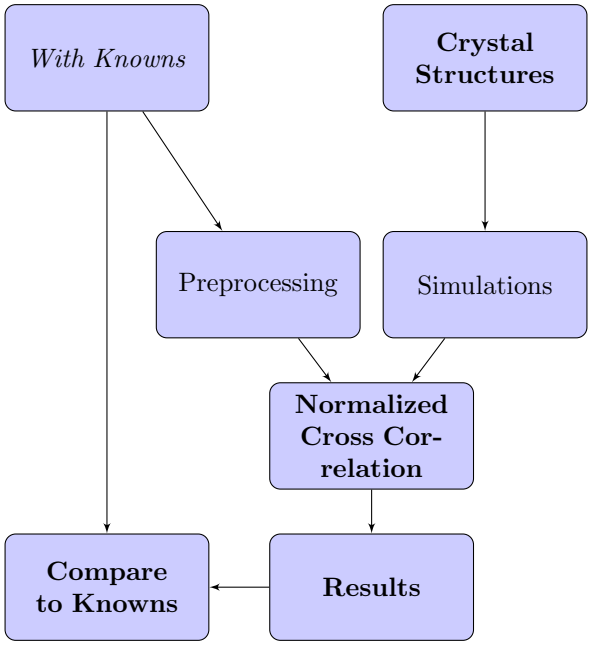


Figure 3.2: Tilt series Data Workflow: Schematic representation of analyzing template matching using '5D' tilt series as input.

4 Results

4.1 Analysis based on simulated data

Simulated data can be used to directly test the method of template matching and evaluate starting parameters without addition deteriorating effects, such as noise, effects present in experimental data. This approach takes advantage of having a known ground truth to compared the template matching results to. This section covers the results from using simulated data, based on kinematic theory described in 2.2.2 as an input, described in 3.4.1, on the parameters of camera length, intensity rescaling and orientation sampling.

4.1.1 Camera Length

The camera length is an important parameter for data and can have a large impact on how reliable template matching can be for a given sample. Camera length controls how much of reciprocal space and how many spots are visible and can effect resultant NCC scores and therefor matching results.

The simulated test data sets are created by taking 4000 random library entries from a library created with a resolution of 0.5°, then given a random rotation in the in-plane(z) direction. This allows for sampling of the entire relevant orientation space as shown in fig 4.1. The templates are given a gaussian blur of sigma 0.5 to better mimic experimental data as the templates are simulated as single point patterns. For this test these artificial data sets are created with varying reciprocal space calibrations,given in $\text{Å}^{-1}/pixel$, controlled experimentally by camera length, and then matched against the corresponding library from which the artificial data was picked.

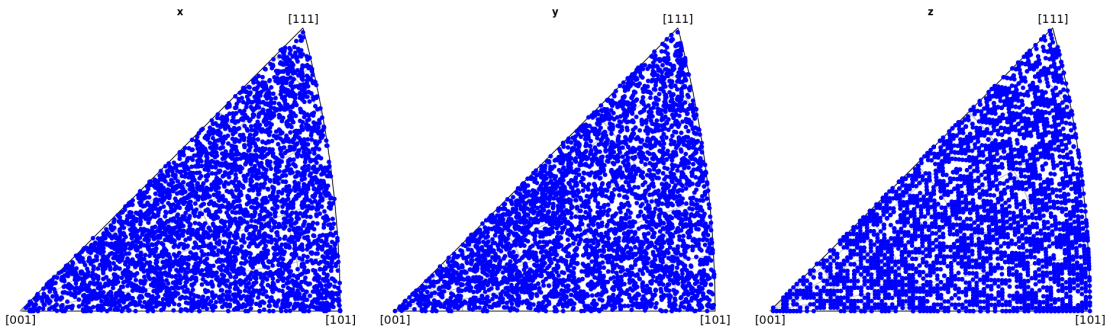


Figure 4.1: IPF view of how the randomly selected orientations sampling the cubic orientation space.

This is done here with Au and Cu, both fcc, to show how the lattice parameter(s) relative to camera length can effect the results and average accuracy. The misorientation between the known orientations used for simulation (ground truth) and the best matched orientations can be used to test how close the template matching gets to the true value. One way to represent these results is by the mean misorientation angle of all 4000 patterns, while removing outliers with over 10° misorientation angle, as in fig 4.2.

Here we see that clearly the mean misorientation angle decreases as calibration increases and for

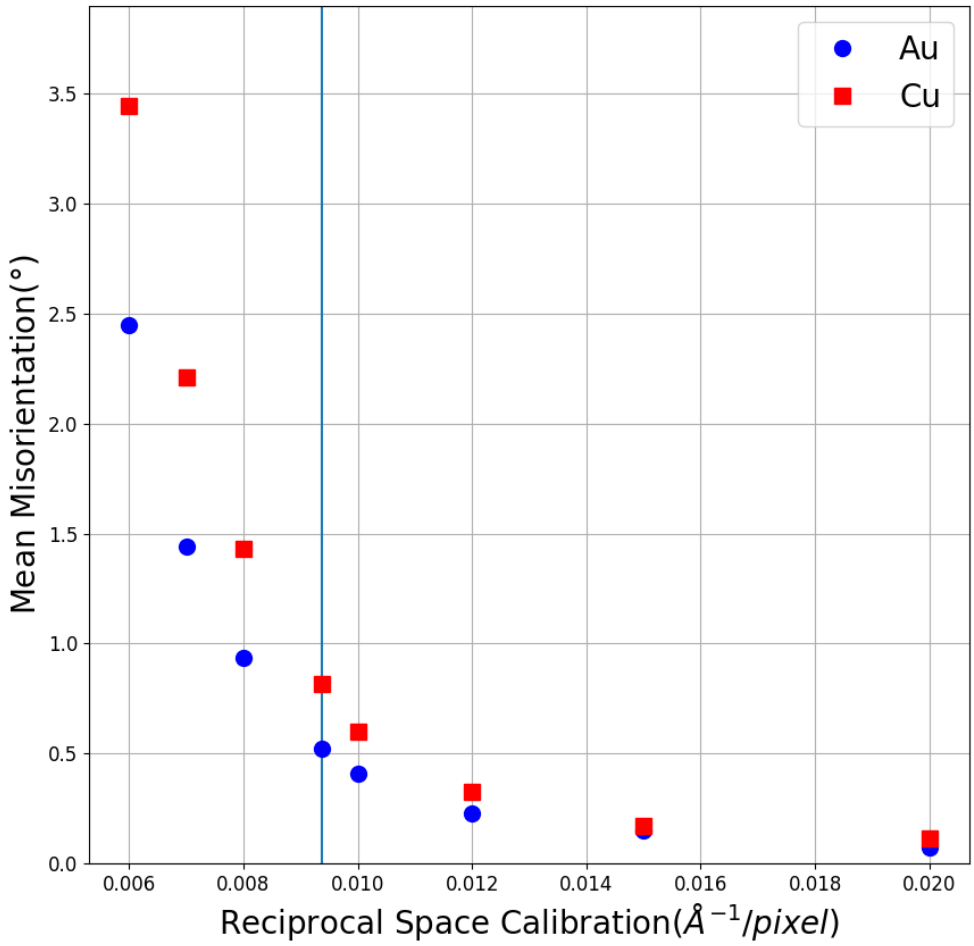


Figure 4.2: Mean misorientation angle between known orientation and template matching result orientation as a function of reciprocal space calibration for Au and Cu. The vertical line represents the experimental camera length of 12cm at 0.00938Å^{-1}/*pixel*

Au the point corresponding to 12 cm camera length is at roughly 0.00938Å^{-1}/*pixel*. It also shows that Cu(red square), which has a slightly smaller lattice parameter than Au(blue circle), gives worse matching results for every point as the diffraction spots are further apart than those in Au. This shows that there is a relation between the number of diffraction spots present in the pattern and accuracy of the template matching.

This can also be represented as the percentage of the patterns that match exactly to their known orientations. Here we see that the number of patterns that match exactly to the known decreases from 99% for both Au and Cu very quickly as the calibration decreases, falling to 44% for Au and 23% for Cu at a camera length of 12 cm.

4.1.2 Intensity Re-scaling

The NCC based template matching approach can be done with different intensity scales for the patterns and templates. By doing so, NCC scores can be effected to produced better or worse results. Here linear, binary and logarithmic intensity profiles are tested to find which one can

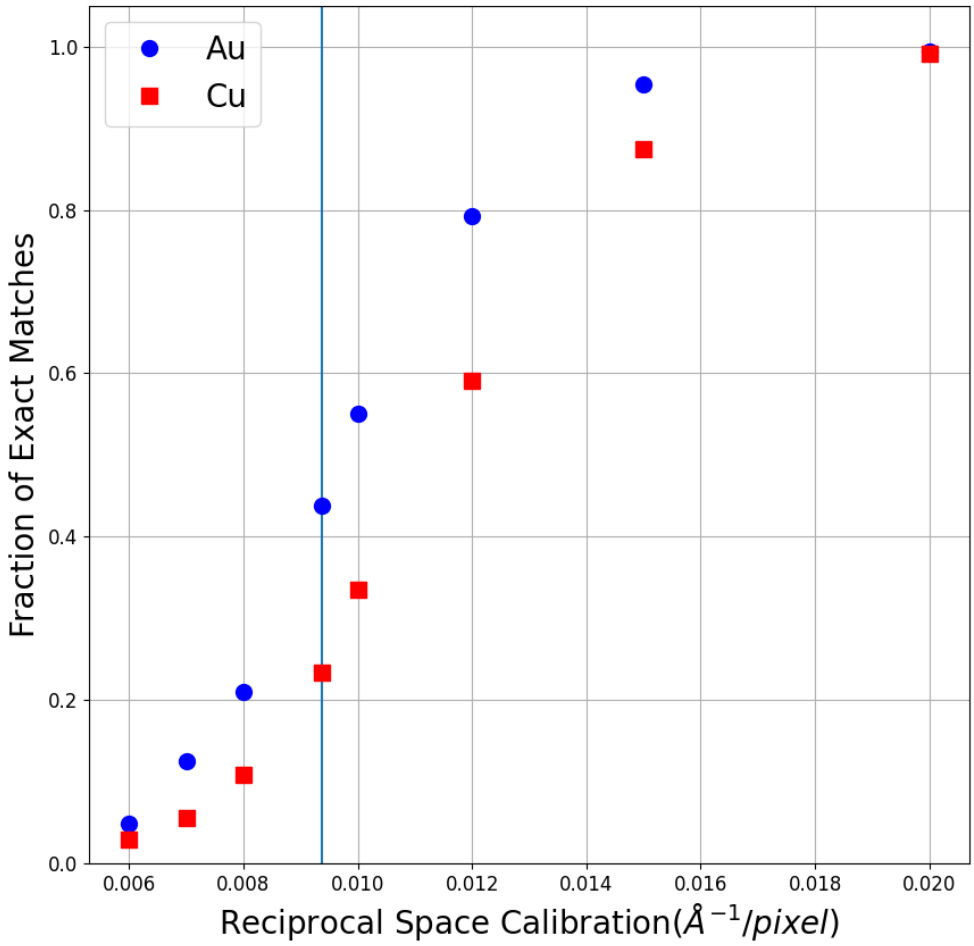


Figure 4.3: Fraction of patterns that are matched exactly(a misorientation of zero) as a function of the reciprocal space calibration. The vertical line represents the experimental camera length of 12cm at $0.00938 Å^{-1}/pixel$

produce better results. Linear is without any changes, binary shifts the intensity scale to be either 1 or 0 and logarithmic takes a logarithm of the original intensity values. The scaling is done on both the pattern and template for consistency. Once again the simulated data is created with 4000 random patterns for a library of 0.5° resolution using Au at 0.01 calibration, as in 4.1.1.

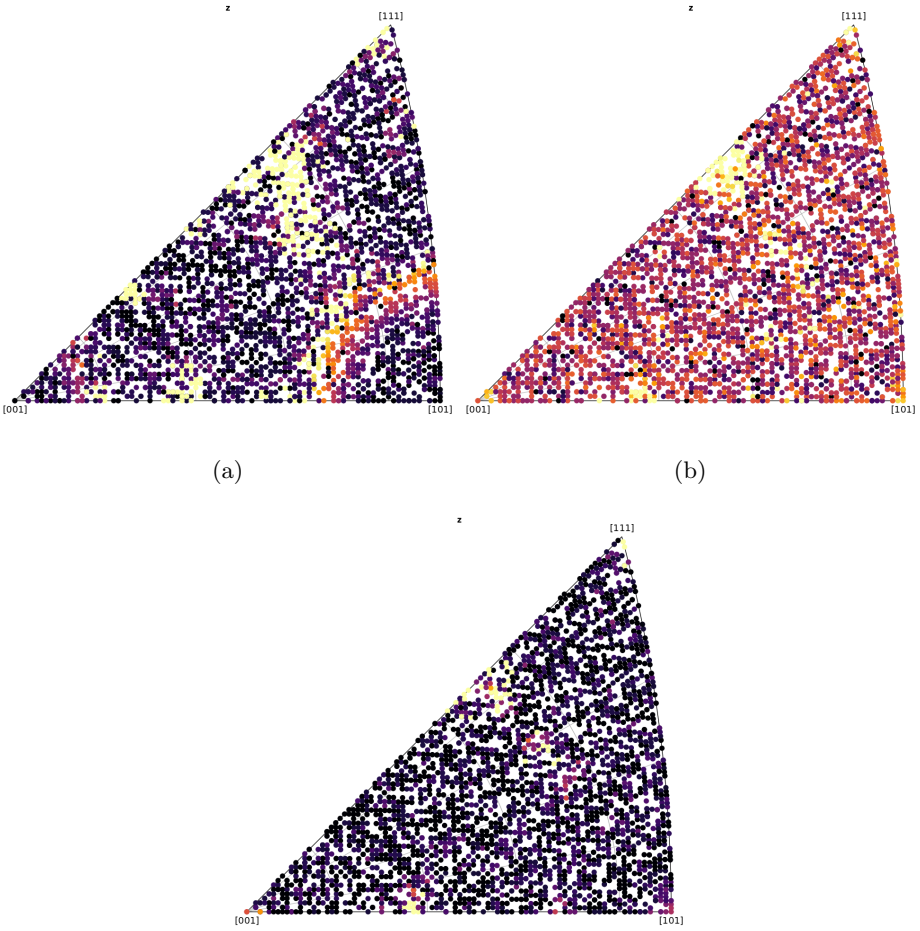
Table 4.1: Effects of re-scaling expressed in NCC score, mean misorientation angle and exact matches for three different intensity scalings.

	NCC score min/max	Mean Misorientation Angle	Percent of Exact Matches
Linear	0.0759/0.1559	0.90°	21.2%
Binary	0.0326/0.0734	2.18°	3.7%
Log_{10}	0.0127/0.0527	0.43°	47.3%

An important observation is that the absolute NCC score is changed by the intensity scale, so it is impossible to compare multiple results with different intensity scales and impossible to say absolutely what a "good" correlation score is. The logarithmic scaling out preforms both linear and binary and binary is particularly poor at matching the inplane angle correctly, as instead of

gaussian peaks the spots are circles of uniform intensity.

The results of intensity scale choice can alternatively be represented in IPFs colored based on the misorientation angle of each individual pattern. From these plots it can be concluded again that the logarithmic scaling is better than the other two and it provides the most uniformly distributed matching results across the varying orientations, for example linear scaling preforms poorly in a quarter circle roughly 10-15°away from the [101].



(a) (b)

(c)

Figure 4.4: Misorientations represented in Z-IPFs of (a) Linear scaling, (b) Binary scaling, (c) Logarithmic scaling

Additionally, logarithmic scaling has multiple additional parameters that can be changed to produce different results.

$$log_b(x + a) - c \tag{23}$$

Here a is a small faction, for example 0.01, that serves to stop infinities, b is the logarithm's base and c is some number whose value is related to the minimum intensity value in a data set. Doing the same procedure but changing the re-scaling by: (b) is tested with 2, 10 and 100, a is varied from 1 to 0.00001 and c is varied from 0 to 0.13 times the minimum intensity.

For the different bases 2, 10 and 100 with an a of 0.1; there is no difference in matching with this data set, the bases having no effect on which pattern is picked. Therefor, at least in this simulated test case, the choice of base appears to be irrelevant.

Varying a while keeping a base of 10 and observing the mean misorientation between ground truth and matching results(figure 4.5) shows extreme difference between 1 and the rest due to a number less than 1 producing a negative weighting to spots present in the template but not in the pattern. It also shows that smaller is not better; 0.01 giving the lowest misorientation.

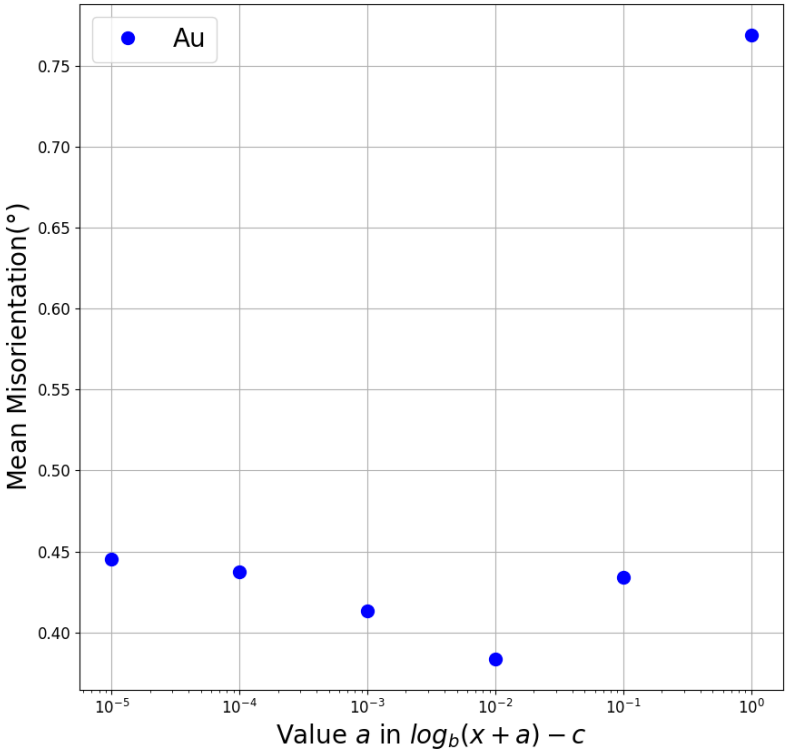


Figure 4.5: Parameter test of a in equation 23, mean misorientation angle as a function of a.

Finally, it is observed from varying c(figure 4.6) that it actually makes the matching worse to use this addition parameter, as well as it having the risk of damaging the data if the number is to large; here, past 0.2 times the minimum intensity, the matching essentially breaks down.

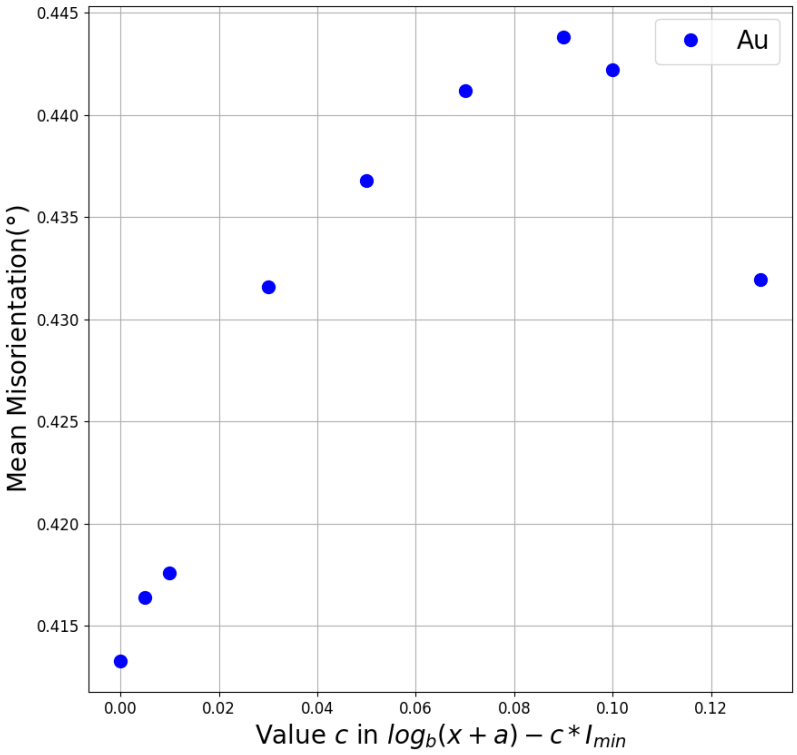


Figure 4.6: Parameter test of c in equation 23, mean misorientation angle as a function of c.

4.1.3 Orientation Library Step Size

As discussed in 2.4; when creating the template library, it is necessary to choose the orientations that will be used to simulate the templates. The resolution and number of these orientations impacts the overall resolution of the matching results: if the matching can only choose from patterns spaced at 5° apart, then the matching can never be better than 5°. However this does not necessarily mean that if the orientations are gridded at 0.05° then the matching resolution is 0.05°as it is not the only factor. The goal for this parameter test based on simulated data is to find the point where the simulations' gridding is not the controlling factor on the resolution of matching, without adding unnecessary computation time as more templates equals more matching time. The setup for this is to create a simulated data set of a random 4000 patterns from a library with a sampling of 0.1°and with gaussian blur, similar to the previous sections, see fig 4.1. This simulated data set is then matched against libraries with sampling of 0.1°, 0.25°, 0.5°, 1°and 2°.

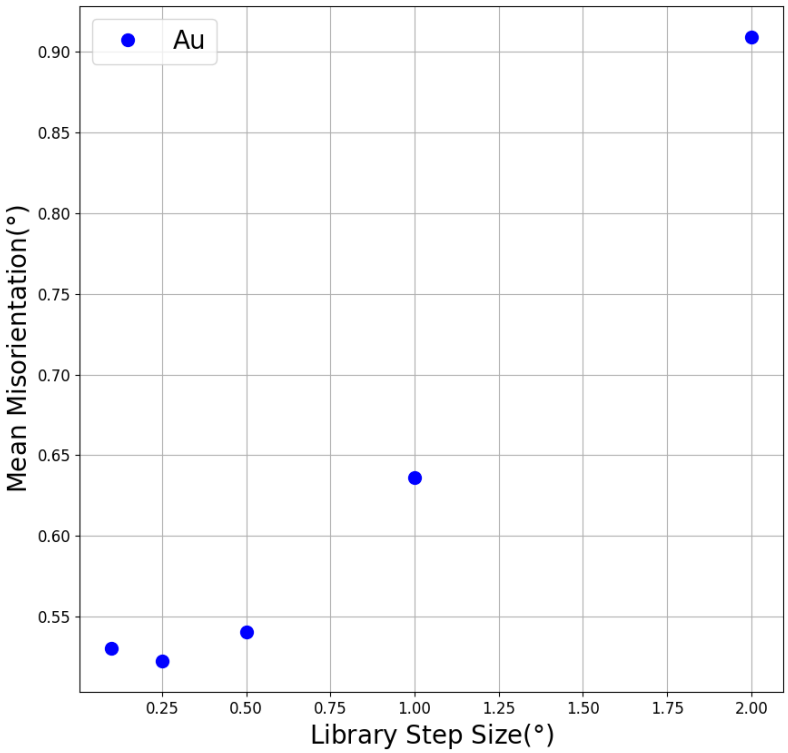


Figure 4.7: Mean misorientation angle between known orientation and template matching result orientation as a function of template library sampling angle.

These results (fig 4.7) show that the mean misorientation decreases as library sampling decreases up to a point, here around 0.5° , before having serious diminishing returns. The mean misorientation even increases from 0.25° to 0.1° , showing that smaller angular resolution libraries are not always better for template matching.

4.2 Preprocessing of Experimental Data

As opposed to the relatively simple simulated data, experimental data introduces additional challenges to accurate template matching. Here the alignment of the patterns and the effects of background subtraction intensity scaling and calibration will be examined. Generally with experimental data, a ground truth is not known, so results will be presented in general terms/qualitative description or based on NCC score.

4.2.1 Alignment

Proper alignment of the data stack is necessary for accurate matching as the templates are simulated with their center at the center of image and matching is done on a pixel-by-pixel basis. Therefor is important that the experimental patterns also have their centers at the center of the image. This step is a necessary step and should always be done, however there are multiple different methods for doing such. These methods, described in 2.4.1, are cross correlate, blur and interpolate. Based on the experimental data sets for AU nanoparticles, Au thin film and CPU specimen data, the following observations are made. The cross correlate method works well for circular direct beams but will do poorly for direct beams that are distorted, it is also a relatively computationally intensive method. The blur method will work even if the direct beam is a bit distorted but will lose efficacy is the center of beam is far away from the center of the image. The interpolate method is more consistent because it is simplistic as it only really looking for the brightest part of the image, however is often lacking because of this. In general, for the experimental data used here, the tilt-shift purity of the microscope was good and shifts required to achieve a aligned data stack of the patterns were small (0.5-1.5 pixel). For the largest scanned area, the CPU specimen, the three methods return shifts of ¡ 3 pixels. This is much smaller than typical beam diameter (15 pixels). Note that in the TM the experimental disc is compared to a simulated spot, therefor alignment need not be 'perfect', however it is still important to verify the stack alignment. No matter which method is used, it is important to only run the centering function on the part of the data stack that contains the direct beam, a square with a half width length of twice the average beam width is used. Further analysis will use the blur method with a sigma of 3.

4.2.2 Background Subtraction

Dealing with background noise is an important part of template matching experimental data based on pattern and pixel intensities. The background noise can muddle to true signal, often making the template matching algorithm match spots in the template to background noise in a pattern. There are four method for background subtraction available in Pyxem, as described in 2.4.1; difference of gaussians, median kernel, radial median and h-dome. Figure 4.8[2] shows the results of these methods on an example experimental pattern from the Au nanoparticles sample. The largest challenge with background subtraction is its subjective nature, however this does not diminish its huge importance on the final matching outcome. One can judge by eye the effect, but the numerical difference in matching score might be different, as shown in fig. 4.2.

Background noise is case dependent, which makes extracting general trends difficult. A quantitative comparison based on NCC score is difficult as the score will depend on the intensity of the background in a given data set and the applied method and degree of background subtraction. Here the four methods have been tested on the three different data sets. The Au nanoparticles have a relatively low background intensity compared to the Au reflection maxima. The CPU, relative thick, has more background between the spots. In general, difference of gaussians seems to be the most robust way and is applied in the further study when using experimental data. To demonstrate the effect of background subtraction, with each of the four methods and without is

[2]Note that these and later DPs are not indexed like those in the theory for two reasons: The point of template matching is that the orientation and indexes of the DPs are unknown and indexes of these patterns are irrelevant to what is being shown, i.e. the effect of background subtraction.

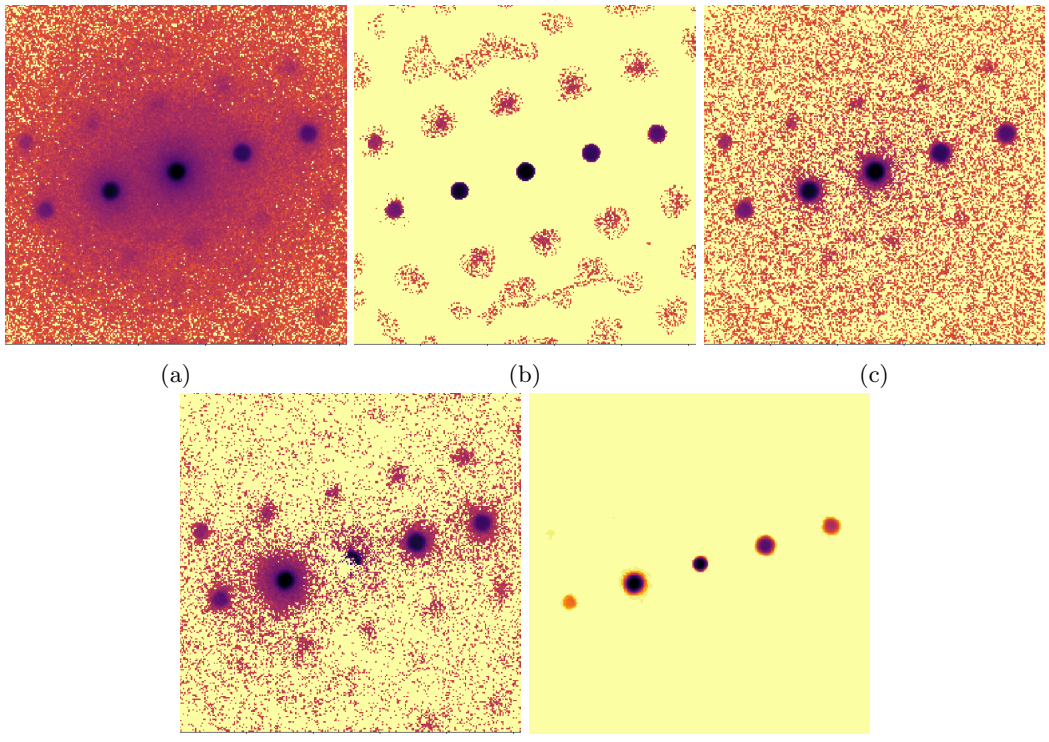


Figure 4.8: Example of an (a) base diffraction pattern for sample A 3.1 and same pattern after (b) difference of gaussians, (c) median kernel, (d) radial median and (e) h-dome.

shown for an example pattern in data set A, see Fig. 4.8 The corresponding NCC score range is given in table 4.2, demonstrating that score is not a good indicator of which method preforms best.

Table 4.2: Normalized cross correlation scores for the best match of the pattern shown in Fig. 4.8 for the base experimental data without background subtraction and the four methods previously introduced.

Base Experimental	Difference of gaussians	Median kernel	Radial median	H-dome
0.00757245	0.00772615	0.00761301	0.0370034	0.01844578

Different samples may exhibit different background effects, thicker samples may be more difficult due to increased dynamical scattering or different materials may have worse single to noise ratios; however for the samples used in this study, difference of gaussians is the preferred method, as it is shown to also work on the thick Cu sample C 4.9.

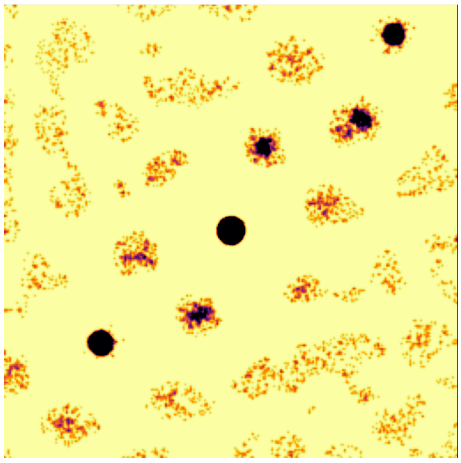
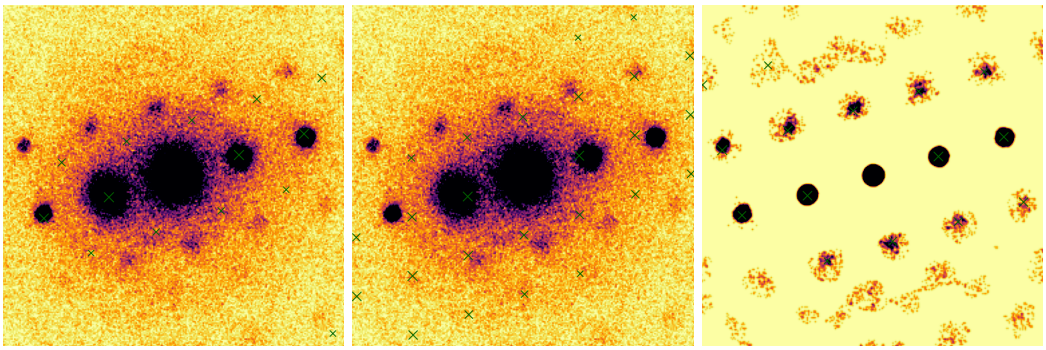


Figure 4.9: Difference of gaussians applied to a single pattern of a thick Cu sample; sample C 3.1.

The background noise, and thereby the method used for subtracting it, has been shown to effect the NCC score and therefor can effect the intensity re-scaling aswell. As shown in 4.1.2, $log_{10}(I+0.01)$ gives the best matching results on simulated data, however it is necessary to assess the efficacy of this when experimental data is matched, particularly when dealing with the effects of background noise.

Figure 4.10 shows that the logarithmic scaling does not produce better results when using data with background noise present. This makes sense as the point of using the logarithmic scaling is to give the weaker intensity spots more weight in the NCC scoring, however this is effectively giving the noise additional weighting, making the results potentially worse. Additionally, the logarithmic scaling punishes the diffraction spots present in the template but not in the experimental pattern, also described in [5].



(a) (b) (c)

Figure 4.10: Effect of the logarithmic scaling, described in 4.1.2, on experimental data. Results with (a) no background reduction and linear scaling, (b) no background reduction and logarithmic scaling and (c) difference of gaussians and logarithmic scaling. Green 'X's indicate the best matched template.

4.2.3 Effect of Calibration

The calibration of the reciprocal space scale, given in $\mathring{A}^{-1}/pixel$, is the final important step before template matching. This is not preprocessing applied to the patterns, but rather information from

the data set that must be given to the simulations in order to properly simulate the prospective patterns. In section 4.1.1 the absolute value of the calibration was investigated, effectively the experimental camera length, however here the relative accuracy to the true value is important. The standard way of finding the calibration is to take data of a standardized sample, normally a fine grain size polycrystalline Au thin film, that creates diffraction rings that can be used to find the pixels to each ring[12]. This could be done at the same time as the sample or done prior to create tabulated values of camera length to calibration values. However the problem arises that this standard might not always be taken with every sample, and the tabulated values might be close but could drift over time or the actual specimen/area studied might be on a slightly different height. As such, having a method to find the calibration from the data set is not necessarily ideal, it is often required. One way of doing this is to simply count the pixels of a known distance, for example for a cubic structure the distance between [220] and $[\bar{2}\bar{2}0]$ should be twice $\sqrt{8}/A$ $\mathring{A}^{-1}$, and from this the calibration can be found. The trouble with this method is that the pattern must already be indexed to know which diffraction spot is which; this might be simple if there is a known zone pattern present, but an additional step if not.

An alternative developed here is to optimize the calibration based on the NCC score. Here it is possible to compare the scores because the scaling and pattern are not changing, only the templates. Doing this allows for a optimal calibration value from a given pattern of any orientation and does not required manual pattern indexing before hand. This gives a objective criteria for finding the calibration value without needing a standard test data set. Figure 4.11 shows the general shape that the NCC score takes as a function of calibration. The peak of this line is often rather broad, showing that the calibration can be slightly different (1-2%) and still give about the same correlation score. This method, for samples or portions of samples where the phase is known, has the advantage of being generally applicable and does not require a pattern to be indexed before hand like the previously mentioned method.

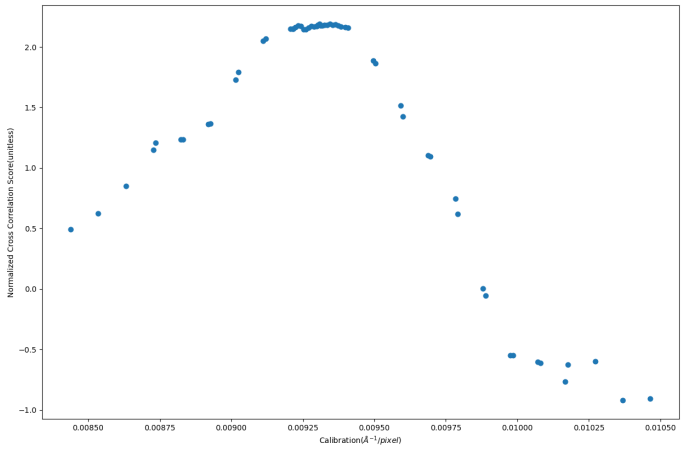


Figure 4.11: Plot showing the changing NCC score as a function of calibration($\mathring{A}^{-1}/pixel$).

4.3 Examining Tilt Series

A tilt series can be created by taking data for a data set then tilting the specimen by some set amount and then taking another data set. Multiple of the data set is then a tilt series, in essence a 5D data set. A tilt series could be a very useful both verify the results of template matching, through an imposed known, and to potentially more about a given sample that a single data set might not include.

4.3.1 Simulated Tilt Series

Before discussing the experimental tilt series, a controlled simulated tilt series was examined. Similar to how the simulated data was created in 4.1, random patterns are chosen from the simulated templates and then rotated by a random in-plane angle. These patterns are then tilted by an known angle(here 5°). These patterns are then template matched as has been described previously with a intensity function of $log_{10}(I + 0.01)$ and a template library step size of 0.5°.

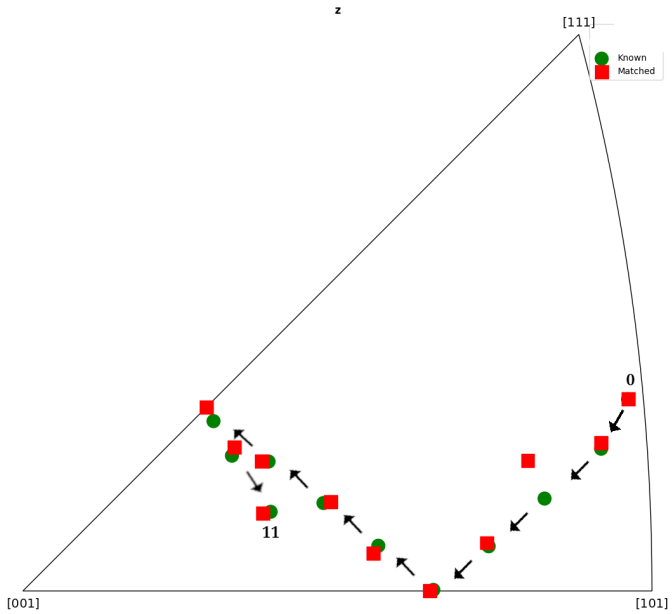


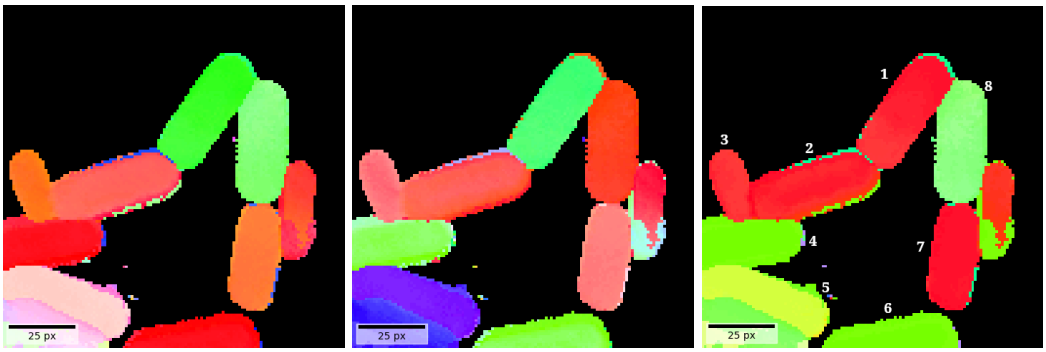
Figure 4.12: Example of simulated tilt series(tilt step 5 degrees in the x direction, template step size 0.5 degrees) of a single pattern with the known orientation (green circles) and template matched orientations (red squares).

From figure 4.12 there are a few general observations that can be made. The second tilt quite mismatched vs the known orientation. Another is how the boundaries of the symmetry reduced zone appear the make the orientation 'reflect' off of them. It is then possible to calculate the misorientation from one tilt to the next, for this tilt set the angles are: [4.74°, 5.03°, 6.26°, 5.19°, 4.°, 4.81°, 5.84°, 5.65°, 3.60°, 5.30°]. This set, Fig. 4.12, has two patterns with the match off by more than 1°, the 3rd and 10th, however often the matching can be off by much more, just like in section 4.1, so misorientations of over 10° from the expected tilt are excluded from the mean. After doing this the mean misorientation between the tilts for the entire simulated data set of starting patterns is found to be 5.18°, with about 10% of misorientations being excluded.

4.3.2 Au Nanoparticles Tilt Series

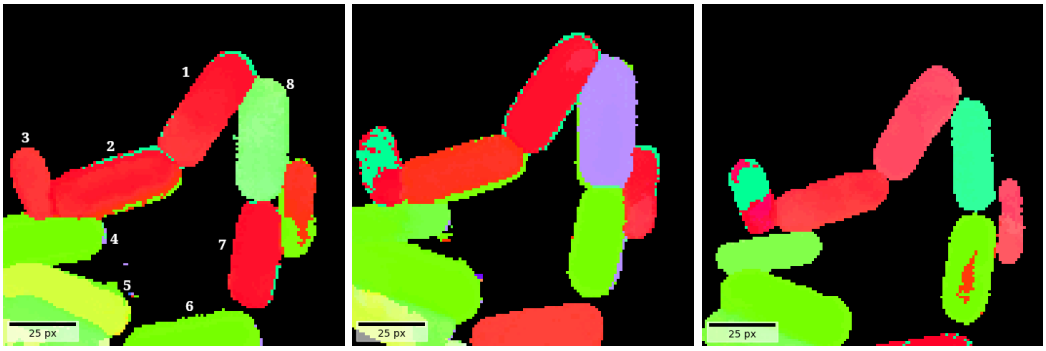
The single axis tilt series of Au nanoparticle data sets (sample A) contain eight clearly visible nanoparticles as can be seen in the constructed orientation maps for the three orthogonal directions. The z-direction is the beam direction (Fig. 4.13.c) that will be used for comparison across the three different tilts. Figure 4.13 shows the selected region at tilts 5/0 for (b) and 9/0 (c) in the z direction.

The misorientation angle between the matched orientation of each tilt can be found and compared to the expected value. It is important to note that the orientation for each particle is found by finding the mean orientation of each particle before comparison. In table 4.3 the misorientations between the tilts is presented, which shows that with only three tilts, the results can be rather unstable; the found tilts seems to vary a lot where the two gonio tilt step sizes was well defined. If there is a mis-indexation in one tilt then difficult to tell which one exact is the problem.



(a) (b) (c)

Figure 4.13: Orientation map of the Au nanoparticle sample at tilt 0/0 from the (a) x, (b) y and (c) z directions.



(a) (b) (c)

Figure 4.14: Orientation map of the Au nanoparticle sample in the z direction in (a) tilt 0/0, (b) tilt 5/0 and (c) tilt 9/0. Coloring is based on the IPF coloring as in Fig. 2.5.c.

Table 4.3: Table of the misorientations between the tilts for each of the 8 nanoparticles in sample A.

Particle Number	Tilt 0/0 - Tilt 5/0	Tilt 5/0 - Tilt 9/0	Tilt 0/0 - Tilt 9/0
1	8.4	6.6	12.2
2	2.2	3.6	2.4
3	9.1	25.3	31.3
4	1.9	29.4	28.5
5	5.9	0.6	6.3
6	27.3	8.2	30.8
7	29.9	0.7	30.0
8	14.0	14.6	6.8

The data set was used as a proof of concept to develop the code for tilt series in conjunction with D, and to test what should be changed or added in the tilt series acquisition or the processing of tilt series. As single orientation maps of randomly oriented nanoparticles the maps look plausible, so the tilt series helps to where the template matching goes poorly.

4.3.3 Cu and Si CPU Tilt Series

The CPU sample is an interesting sample as it contains Σ3 twin within the Cu band allowing for the known of the twin misorientation along with the known tilt angle. Data from a portion of the crystalline Si substrate was also taken as a reference starting at the [101] in the tilt 0/0. This

reference is used for calibration and as well as serving as a reference for the tilt angle for each step, as the patterns can be used to create sum patterns with very high signal to noise ratios and Si having a larger lattice parameter than Cu that allows for more accurate template matching, as shown in 4.1. Note that theis tilt series contains five tilts taken at the two perpendicular gonio axis, called x and y for simplicity, compared to the single axis tilt series with three tilts in 4.2. That series also does not contain a known either in the form of a reference (Si substrate) or in the form of a known orientation relation (twin).

From figure 4.15, it is observed that the twins are at roughly the same orientation([101]) relative to the Z direction, translating to the patterns are only rotated in the inplane angle by 60°. By calculating the mean orientation for each side of the twins and finding the misorientation between them the results in table 4.4, are found. This shows that all but the first tilt are within 1° of the expected 60° misorientation and hence that the matching of each single or summed pattern on both sides of the twin plane was reasonable close to the actual orientation.

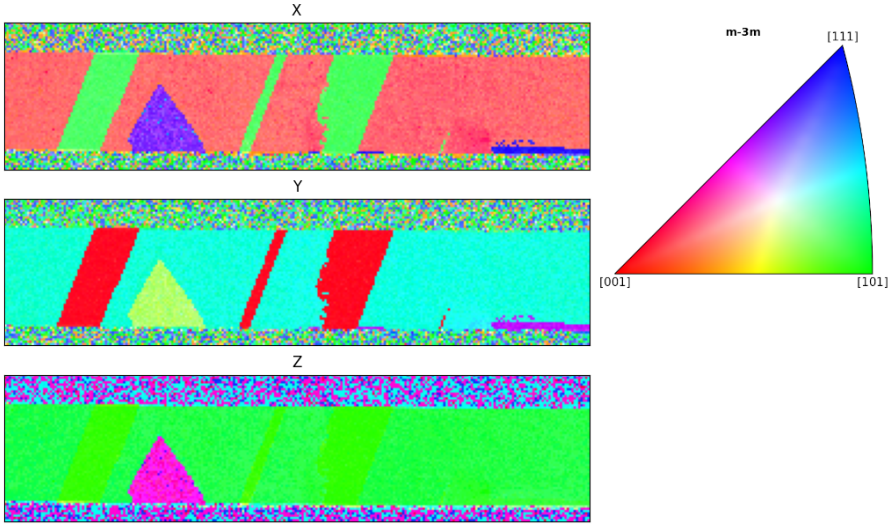


Figure 4.15: Orientation map of the Cu band at the tilt 0/0, colored using the IPF color scheme described in section 2.1.5.

Table 4.4: Calculated misorientation across the Σ3 twin plane for the five different tilts in the CPU sample tilt series.

Tilt 0/0	Tilt 5/0	Tilt 10/0	Tilt 0/5	Tilt 0/10
58.4°	59.2°	59.3°	59.8°	59.5°

It is also possible to find the misorientation between the tilts for each side of the twin plane. Table 4.5 shows that the Si agrees with an experimental tilt of 5+-1°, the y tilts also agree, tilt 0/5 - tilt 0/10 underestimating the misorientation angle due reflecting off of the border of the reduced symmetry area, further discussed in 5.3. However the x tilts are far outside the expected range but the Si reference shows that it is not an experimental error but a systematic error with the template matching. So the relative, within one tilt position, represented by the relative orientation between the two grains, is close to the expected value for a Σ3 twin. Where the absolute orientation, represented by tilt of each grain, under represents the tilt when tilted along the x direct.

4.3.4 Au Thin Film Tilt Series

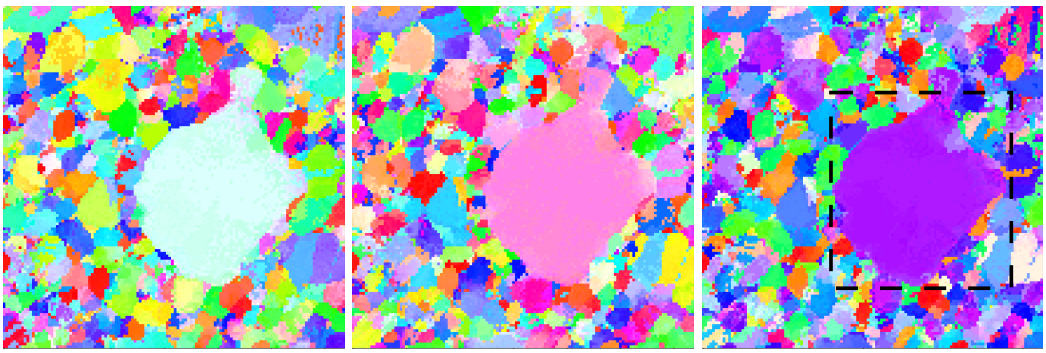
The final sample is a poly-crystalline Au thin film with the particularly interesting feature being a large grain on the [112] zone. Figures 4.16 and 4.17 shows the tilt 0/0 and tilt 0/10 in the x, y and z directions. Just from these images the complexity of the data is apparent, there many smaller grains and even grains with visible twinning. However, all of these small grains mean plenty

Table 4.5: Calculated misorientation between the given twin for the Si substrate and the two difference side of the Σ3 twin plane.

	Si Substrate	Twin A	Twin B
Tilt 0/0 - Tilt 5/0	4.2°	2.2°	2.5°
Tilt 5/0 - Tilt 10/0	4.5°	2.8°	2.1°
Tilt 0/0 - Tilt 0/5	4.9°	5.8°	4.6°
Tilt 0/5 - Tilt 0/10	4.2°	3.4°	3.7°

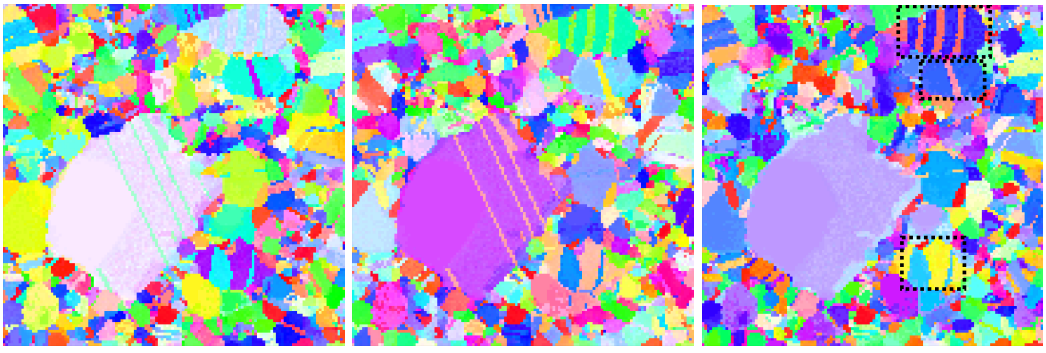
of overlapping patterns, which is one of the main challenges present with template matching; an experimental pattern which is composed of multiple different DPs is matched to single template. In the further these areas and the issue of overlap will be ignored.

weaknesses of template matching. Another observation is that there are twin planes present in the large grain, however they are not visible in the tilt 0/0 as the Σ3 is 'invisible' when viewed along the [112], in essence both sides of the twin appear to be identical in the constructed orientation maps[3]. However from tilt 0/10 there are visible, even if small, twins present. There are other grains present that appear to have twins in them as can be seen in Fig. 4.17, for different directions. The contrast of twins in a grain, visible as a pattern with parallel straight lines can also seen in the medium sized grains, marked with dotted areas in 4.17.



(a) (b) (c)

Figure 4.16: Au thin film orientation map of the tilt 0/0 viewed from the (a) x, (b) y and (c) z directions with main grain of interest marked by dotted outline.



(a) (b) (c)

Figure 4.17: Au thin film orientation map of the tilt 0/10 viewed from the (a) x, (b) y and (c) z directions with smaller grains with apparent twinning marked by dotted outlines.

Once again the misorientation between the tilts can be calculated as done previously, here for the large grain that is at [112] in tilt 0/0. Table 4.6 shows that the tilt 0/0 - tilt 5/0 and tilt 0/0 - tilt 0/5 are within 1° of the expected 5°. However the other tilts are far off of this value, so there is a clear problem here.

[3]This is due to there being an inherent 180° ambiguity along the z (optic) axis.

Table 4.6: Misorientation angles between the given tilts for the large grain in the Au thin film data sets.

	Misorientation Angle
Tilt 0/0 - Tilt 5/0	5.7°
Tilt 5/0 - Tilt 10/0	48.4°
Tilt 0/0 - Tilt 10/0	52.7°
Tilt 0/0 - Tilt 0/5	4.0°
Tilt 0/5 - Tilt 0/10	41.2°
Tilt 0/0 - Tilt 0/10	43.9°

5 Discussion

5.1 Analysis based on simulated data

Utilizing simulated data to run parameter tests in controlled ways with known ground truths is useful way of examining the efficacy of the template matching routine, for specific process parameters. The presented results can be further compared to TM results from experimental data, to acquire more accurate and reliable answers. Hence, the here created python notebooks D,C should be of interest beyond the specific examples discussed in the present work.

The method used for creating simulated data sets allows for randomized sampling of orientation space and across the range of euler angles. This is more robust than simply using the simulated templates themselves because, as described in 2.4.3, the simulations do not sample the first euler angle (in-plane rotation angle). It also allows the data set size to remain constant, as when simulating cubic templates at 1° there are roughly 1000s template but at 0.1° there are over 100000.

The way in which the accuracy is compared is the misorientation angle between the found result and the known ground truth. This is superior to simply comparing the index and giving a right or wrong; as it also shows how close or far a wrong answer was. Throughout the given template matching examples it becomes clear that the NCC score can be a poor judge when comparing difference data sets or the same data set under different preprocessing conditions. When the mean misorientation angle is calculated the obvious outliers are removed, taking into account that misorientation space is not evenly distributed. These outliers are answers that are more than 10° off, this is done to ensure that the mean is a statistical mean and is not affected by answers that are multiple standard deviations away. This extremely high misorientation angles are mostly a result of patterns near zones of high symmetry, [112] and [123] being the most common, observed in 4.4. These patterns have strong systematic rows and many patterns near the zone are extremely similar to the on zone pattern, cause the patterns to be very unique which is the principle way that template matching can differentiate between patterns.

The choice of intensity scaling function is shown to greatly effect the matching results. The base line linear scale is out preformed by the logarithmic scale, and the binary scale does very poorly especially when it comes to determining the in-plane angle. Binary scaling, also introduced by Cautaerts et. all [5], of the experimental pattern could potentially be used it the disks are smaller, like with unprecessed data, however logarithmic scaling still has the benefit of increasing the relative weight of weaker diffraction spots, but can also punish the extra spots present in the template. The logarithmic scaling is not a new idea as it is discussed by Cautaerts et. all [5], however the work presented here is a more quantitative look at the scaling. It is important to note however that these tests are done on simulated data sets with no background noise, the introduction of which could reduce the efficacy of the logarithmic scaling and it is therefor important to keep in mind the background in experimental data. In conclusion, based on these parameter tests on simulated data sets, the ideal intensity re-scale function is $log_10(x + 0.01)$. This function helps value weaker, often far away from the central beam, spots that have more orientation information as well as when combined with a background of zero, punishes spots that exist in the prospective template but not in the pattern itself.

The important take away from the camera length test is that more visible diffraction spots leads to more accurate matching. This seems quite logical as the more higher order spots with lots orientation present means that patterns are more unique which in turn means that the template matching is more accurate and can discern smaller angle difference better patterns. However while with simulations lowering the camera length has little negative effect, experimentally this is not the case. The diffraction spots far away from the central beam have lower chance of diffraction and thus a poorer signal to noise ratio.

It is only since use of better recording systems [5] and [14] that more spots farther from the center can be captured. This means these diffraction discs are much dimmer and can easily be lost in the

background noise. It is also important to note that since with SPED the diffraction spots are more disc like with a size(radius 2θ) if spots become too close it might become impossible to discern between similarly spaced spots. This has even more an impact if the sample if multi-phase where the d-spacings are similar. If the camera length is too small, the different phases can also become impossible to discern as the patterns of one phase are too similar to the patterns of the other.

Results from the library step size test show that larger libraries with smaller step sizes do not necessarily equate to better matching. As shown in the camera length test, the diffraction patterns have a 'uniqueness', based on the phase, camera length and potentially other experimental parameter, that controls how angular resolution of the orientation mapping. For example, here with Au at 12 cm, the angular resolution is at about 0.5° - 0.55° , past that the patterns simply are not unique enough to tell apart. This is logical as the spots near the central beam, low order, are shared between more patterns as compared to spots far away from the central beam, high order. This reinforces the conclusion that for accurate and reliable matching, patterns should contain many diffraction spots and in particular higher order ones.

An important observation that can be made from these tests is that template matching's accuracy is uneven across orientation space. In figure 4.4, the misorientations are not entirely random; there are concentrated sections with higher than average misorientation. The section around the [112], see fig. 5.1 for pattern, is present no matter which intensity scale is used. The logarithmic scaling does help limit these sections, making those that are present smaller and even completely removing some: the rings centered on the [101] and [001] and the cluster around the [113]. The [001], [101] and [111] as exhibit this behavior to a lesser degree, particularly the [111], where pattern seem to be "pulled" towards certain templates near the poles that give high scores.

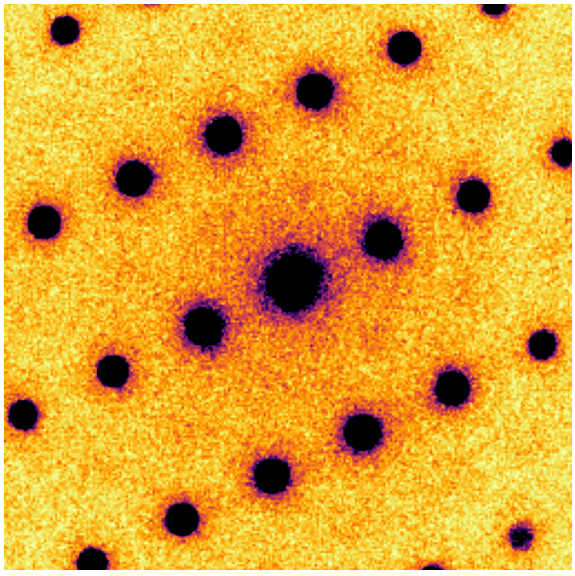


Figure 5.1: [112] zone diffraction pattern representative of the large grain in Au thin film sample.

The results from the intensity re-scaling also show that while the relative values of the NCC score can be used to judge which template matches best, at least in principle as there is the potential for the highest score match to be a misindexation, the absolute value of the NCC score has very little useful information. As is 4.1, the value of the NCC score varies wildly between intensity scales, the linear scale have a maximum score three times that the logarithmic scale, even though the matching results of the logarithm are better. This is important to keep in mind when attempting to draw conclusions from the absolute value of the NCC score itself, for example deciding if a result is reliable based on the score or comparing two different template matching results.

Something not fully explored here is the effect of the inplane angle on the matching results. By rotating a pattern, diffraction spots can be cut off or revealed, either giving fewer spots to match and thus worse matching or more spots and thus better matching. This can also effect the absolute value of the NCC score, observed in experimental data 5.3.

5.2 Preprocessing of Experimental Data

Preprocessing is a very important step that must be done before template matching. As mentioned previously, centering is simply required; the diffraction patterns all need to have the same center and is easiest the make it the center of image. The simulations are done to match this. The best method might vary from data set to data set based on the shape of the central beam on its intensities, however most of the time any of them should work, however it is important to restrict the area that it "looks" for the center to a small part of the images where the central beam is. This ensures faster computation times but also protects against the function falsely choosing bright low order spots instead of the direct beam. The blur and interpolate are both very similar and will often give very similar results while the cross correlate method can often be rather unstable based on the parameters given. The other advantage of blur and interpolate is in computation time, these methods taking 10 seconds, while the cross correlate can take 2 minutes on the relatively standard computer used. For these reasons the blur method is used for the data sets presented.

The general preprocessing is analyzed here, but one common preprocessing step, distortion correction, was not part of the present study. The images could have distortions like shears or stretches that could effect the template matching, however the data sets presented appear to have little to no distortions. Even if they do have small distortions the radius of the spots gives enough margin of error that the matching is still good. Still, if there are non-negligible distortions present in a data set, this would become a necessary step. Based on the inspected data sets, the current experimental set-up has, for the used camera lengths and relatively scan areas negligible distortions. There are distortion correction routines included in Pyxem [12].

This can be done by taking a standard Au grating to get circles of known distance and if there are distortions present the circles will be deformed and would need to be corrected for with an affine transformation [8]. It could be possible to expand the calibration method presented here in section 4.2.3 to also optimize for any changes due to distortions, however it becomes substantially more complex as an affine transformation consists of nine numbers(although for this case only five should be relevant) on top of the calibration which would need to be optimized all together. This would call for a smarter way of optimization than presented here for just calibration, however the principle of optimizing the NCC score would remain the same.

Background subtraction is a very important but also challenging part of template matching as it is case specific. Here it is shown that some diffraction patterns will not match correctly without background subtraction, thus it is a required step. However unlike centering and calibration, it is very subjective and case-dependent. The method of difference of gaussians is shown to be applicable on both thick and thin samples and preforms better than the other methods available in pyxem, section 4.2.2, however the method has two parameters: the minimum and maximum sigmas of the gaussians. These, along with values for thresholding the intensities and gaussian blurring, create range of many possible values that could be used for background subtraction, some of which might be obviously poor choices, however there will still be a range that produces 'good' results but will ultimately be the choice of individual. Cautaerts et al. [5] comes to the same conclusions, as outlined: *'the difference of gaussians method is used and a small gaussian blur is added that helps smooth the weak peaks'*. However the fact is that background noise and therefor background subtraction is very case dependent. This also makes it difficult to suggest optimal values as the values used here on these three samples might produce poor results for another sample taken on the different microscope under different conditions. This challenge is not easy to solve, however what is present here should give some idea of the procedure to get and examples of background subtraction that produce good template matching results.

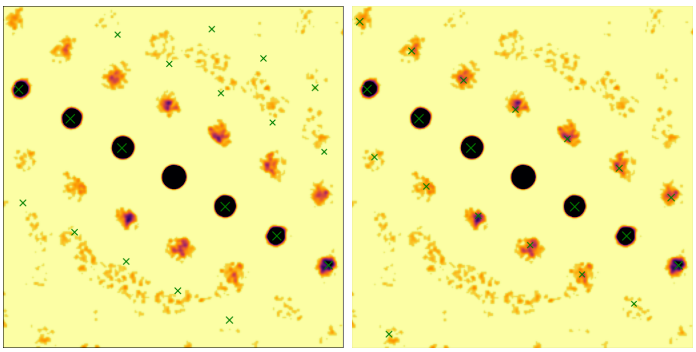
The choice of intensity scaling, also shown on the simulated data in section 4.1.2, has a large effect on the matching results. The most important difference here for experimental data is the background noise present and how that changes the effect of intensity scaling. When using logarithmic re-scaling, background subtraction becomes necessary, as to not give increased weight to the background noise. Thus for optimal template matching, both logarthmic re-scaling, as described in section 4.1.2 and background subtraction from 4.2.2 should be used.

5.3 Examining Tilt Series

Up to now simulations (5.1) and processing single patterns (5.2) have been analyzed. One of the important questions for this study was if taking a series of data sets of a single sample can have added value on evaluating the template matching workflow. Here simulated tilt series can be used to test template matching just like the regular simulated data set presented in section 4.1. It can also do things that would be difficult or impossible experimentally; a sample can not normally be experimentally tilted 50° in one direction without the real space part of the data becoming extremely distorted, increasing in thickness and causing overlap. The simulated tilt series also further backup the observation made previously that template matching accuracy is not even distributed over orientation space, some patterns are just more difficult to match than others. The relative orientation of the tilt angle thereby becomes an important factor. These test can further show that the tilt series should be relatively large angles; the matching simply does not have the angular resolution for a tilt series of 1° tilts to be worth the amount of data that would be collected. Utilizing tilt series has the ability to introduce two different experimental references for verification of the template matching results: i) the tilt step, which is limited by accuracy an linearity of the holder in the microscope as is case with tomography [28] and ii) a reference area, here the [111] Si substrate in the CPU specimen.

An idea that was only partial explored was to optimize the matching the results by searching through more the just the best matched template and finding which patterns return the closest tilt misorientation to the the expected tilt. This has potential to improve the results by taking into account a known(tilt angle) and using the additional information contained in the matches subsequent to the best. This is however quite a complex procedure and if done wrong could to more harm than good, select incorrect patterns for various reasons; such as the misorientation angle being under-calculated when a pattern is tilted over a boundary of the symmetry reduced zone, or when used on experimental data as the tilt angle has a experimental variance.

The Au nanoparticle tilt series is an interest test case. The particles are not completely randomly orientated, due to their shape there is almost always a systematic row present(here either close to a [001] or [101]). The tilt misorientations, table 4.3 show that the matching results can be rather unstable under these conditions. With only three different tilts however, it is often difficult to know under which tilt a particle is misindexed but it is possible. For example, using images in figure 4.14 and the values from table 4.3 to inspect particle 8, the observation that particle 8 is misindexed in tilt 5/0. Sure enough if the matched template is inspected, fig.5.2.a, the pattern appears to be misindexed. However if the list of matches is examine closer the 5th match appear to be a better match, backing up the potential for using the known tilts in the tilt series to optimize the matching results by searching through the list of matches.



(a) (b)

Figure 5.2: Diffraction pattern from sample A, particle 8 with the green 'X' indicating the template of (a) best match and (b) 5th match.

From the CPU sample there are quite a few interesting observations that can be made. The first being that the x tilts(tilt 5/0 and tilt 10/0) appear to have a systematic error in the template matching, table 4.5. Plotting the orientations in z-IPFs, figure 5.3, the slightly low misorientation

angle present in tilt 0/5 - tilt 0/10 can be explained by that fact that the tilt 0/10 orientation gets reflected back into the symmetry reduced zone after passing the boundary which causes the misorientation to be calculated lower than path that it travels, sketched in the figure. Another interesting observation from this is that the twins do not start exactly on the [101] zone.

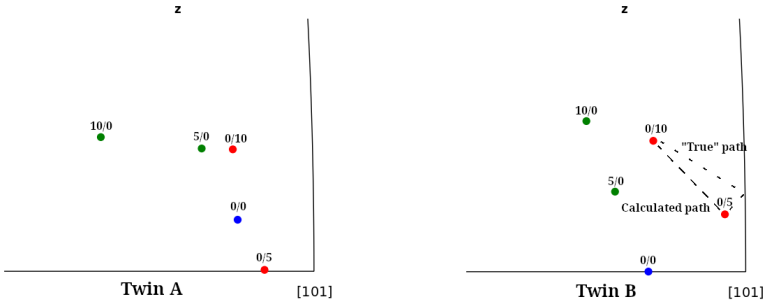


Figure 5.3: Five tilts of the Cu twins represented in z-IPFs.

The Au thin film is a very interesting sample, with a vast amount of data present. Focusing on the main large grain it was observed that the tilt 10/0 and tilt 0/10 had very large misorientations between the others. However, in this case the misorientations might not tell the entire story. Plotting the orientations in an ipf, fig. 5.4, the orientations appear to make some sense. This is evidence that there is more happening at the [112] maybe was first observed. It might not be that patterns match poorly near the [112], but when calculating the misorientations or orientations near the there is some systematic error or this is a quirk of reduced orientation space. It is unfortunate that the twins present with the large grain are too small match and compare as they contain overlapping patterns, however the smaller grains also appear to contain twins, however when the grains that have less overlap are examined the angle between the different sides of the twin is found to be 45°. This is not the expected 60° that a $\Sigma 3$ twin should exhibit. This could easily be because of overlap or multiple twins present in the diffraction patterns as these twins are small compared to the grain size.

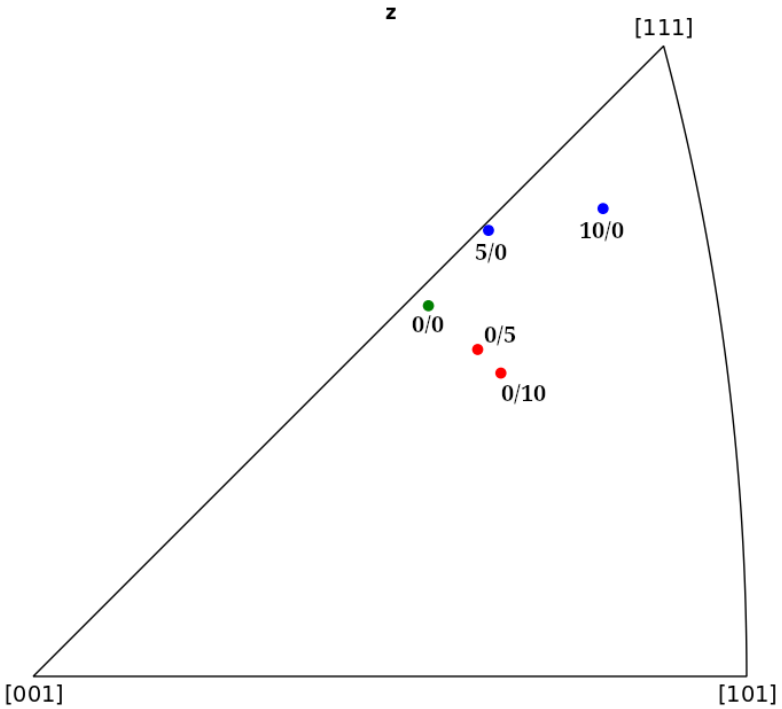


Figure 5.4: Orientation of the 5 tilts for the Au thin film data sets plotted in the Z-ipf

Verification of the template matching results can be done without knowns present the a data set. The most common ways for doing this are the correlation score, although as previously discussed it should be the *relative* NCC score, and reliability [23]. Using the correlation score can work under the right conditions; however as it has been discussed already, the correlation score varies based on the intensities and so patterns with plenty of diffraction spots with naturally have higher correlation scores than dimmer patterns. This does not necessarily mean that higher NCC scores translate to more accurate and vice versa. Something that the correlation can be useful for is sample that vacuum (or bare substrate), because the relative correlation score will be much lower for that part of the data, and can therefor be thresholded as not matching.

The reliability metric [23], given by the difference is correlation score between the best match (first) and the second match. If this is low then the first and second matches are very close in correlation score, however this is very often the case because the second match is often directly adjacent to the first match. This metric also only takes into account the difference between first and second but more matches could be used to improve upon this. This metric can show the edges of grains however so it is not without use conditions.

A new metric proposed is to use relative correlation score to normalize the misorientation angle between the first match and subsequent matches and then take the mean so that each point in real space has a number for this metric. This is described by equation 24:

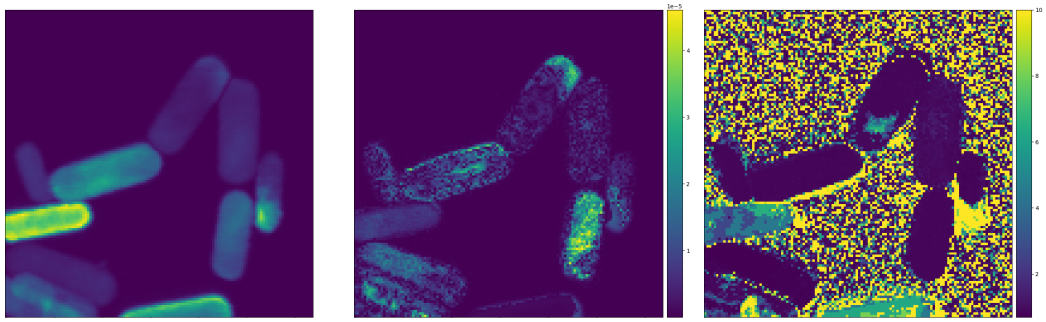
$$\frac{\sum_N (Q_0 - Q_N)(\frac{C_N}{C_0})}{N} \tag{24}$$

Here $(Q_0 - Q_N)$ represents the misorientation angle between first match and the Nth match while C_N and C_0 and the NCC score for the Nth match and first match respectively. This metric has a few advantages. One is that this allows for the utilization of more than just the first and second matches, as the list of matches does contain information it is just not necessarily easy to parse. The aim of weighting the misorientation angle with the relative correlation score is to allow this metric to show the difference between a high misorientation angle with a high relative correlation score, which is likely indicative of poor matching, and a high misorientation angle with a low relative correlation score, which does not necessary indicate poor matching.

The following observations from the three different metrics from figures 5.6, 5.7. The correlation score can be good for visualizing grains and grain boundaries. The reliability index appears to not give as much information as it does for ASTAR [19], most likely due to how ASTAR reports the sequence of matches (however due to ASTAR being proprietary this can not be confirmed). The proposed normalized misorientation angle can show areas of potential misindexation, the [112] grain and plenty of overlap in the Au thin film data or particular nanoparticles in the Au nanoparticle data. It is however 'grainier' or pixelated than the correlation score, the substrate of the Au nanoparticles for intense is mostly homogenous for the correlation score but the normalized misorientation angle varies even the patterns outside the nanoparticles certainly do not match well. So utilizing both the correlation score for thresholding out the vacuum or substrate and normalized misorientation angle can be beneficial analysing how much the returned best match from template matching can be trusted.

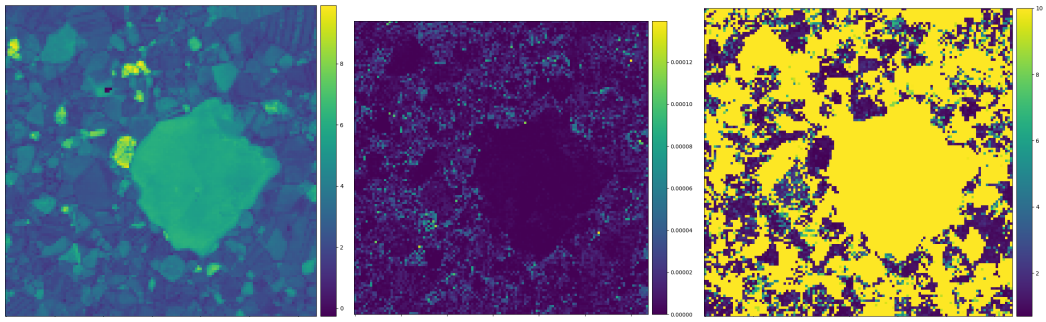
5.4 Optimized Setup

From all of the results and the observations and discussion, an optimum setup to receive the best possible template matching results will be presented. Potentially the most important part of accurate template matching is having as many diffraction spots present in the patterns as possible to determine the best fitting template. As shown in section 4.1.1, both the camera length and the material impact the accuracy of template matching. The smaller the camera length, the more of reciprocal space is capture in each diffraction pattern. This leads to having more unique patterns, as the higher order diffraction spots, those far away from the direct beam, have more orientation information. The material(s) used, or more precisely the lattice parameter(s) of the material(s), has the same effect; larger lattice parameters translate to more diffraction spots present the final



(a) (b) (c)

Figure 5.5: Au nanoparticles at tilt 0/0 (a) correlation map (b) reliability map and (c) proposed normalized misorientation angle map.



(a) (b) (c)

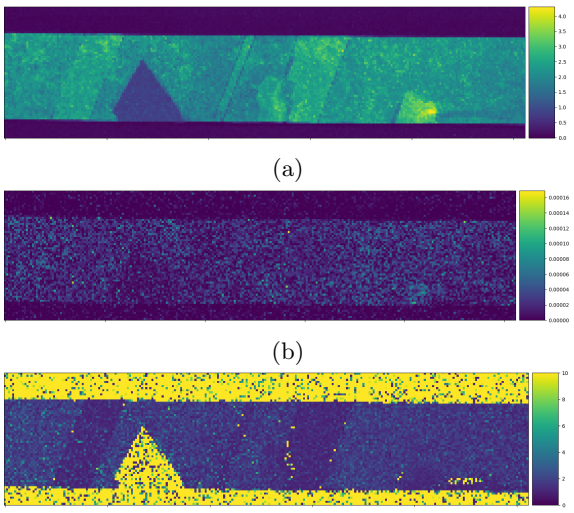
Figure 5.6: Au thin film tilt 0/0 (a) correlation map (b) reliability map and (c) proposed normalized misorientation angle map.

patterns. This is however entirely on the experimental side of the equation. If the material of interest is Cu for instance, a small lattice parameter is simply something that must be taken into account. The camera length will also have an experimental optimum range that is not tested here. With this in mind the smallest camera length that still gives good signal to noise and peaks that are far enough apart should be used when taking data.

The intensity re-scaling is another important factor, 4.1.2. Here it has been presented that a logarithmic intensity scale will produce the best template matching results, in particular $log_{10}(I + 0.01)$ found in 4.1.2. This is due to give a higher relative contribution to the NCC score from the weaker, often higher order diffraction spots. It also helps to give a negative contribution to spots present in the template but not in the pattern when the intensity outside of the diffraction spots is zero.

The pre-processing utilized on the data sets can have a large effect on the outcome of template matching. Centering is simply required with the optimum method being somewhat case dependent, however blur seems to be quite stable. It is important to limit the field of the patterns that the centering algorithm works on to insure that the direct beam is chosen for the center. Background subtraction plays a large role in the accuracy of template matching, and is required when using a logarithmic intensity scaling, 4.2.2. The method of difference of gaussians appears to have to most potential for 'good' background subtraction; removing the noise without removing weak diffraction spots. It is however still important to remember that this is quite subjective and case dependent, so proper choice of parameters is essential. The workflow using jupyter notebooks, makes the processing transparent and used settings and values are documented.

It is proposed and shown how a tilt series can help ensure better template matching results by both providing a known to test results against and identify mis-indexation, as well as measuring the same sample multiple times which can improve both precision and accuracy.



(a)

(b)

(c)

Figure 5.7: Cu CPU band at tilt 0/0 (a) correlation map (b) reliability map and (c) proposed normalized misorientation angle map.

Finally, multiple ways of verifying the results of template matching have been shown: relative correlation score, use of the knowns present in the experimental data and correlation normalized misorientation angle. Verification of results can be very important as many of the parameters in the pre-processing and template simulations steps can greatly effect the final template matching results.

A demo jupyter notebook, C, that will also be used at a workshop given by the TEM Gemini Centre shows the principle steps outlined here. Post-processing is a crucial step in the whole approach of template matching and open-source libraries, like Pyxem, allow for transparency and systematically improving and sharing best practice.

6 Conclusion

The effects of camera length, intensity re-scaling, pre-processing steps, template library step size, calibration and use of tilt series on template-based orientation analysis within Pyxem has been presented. Concrete observations and suggestions about and for obtaining more precise and accurate template matching results have been made.

The conclusions from the simulated data set tests are:

- Experimental camera length should be as small are reasonable possible to capture the most amount of diffraction spots in the images without making them too close together, ideally the diffraction discs have a radius of 3-5 pixel, to be able to differentiate them.
- Intensity scaling should be done on a logarithmic scale to given a higher normalized cross correlation score contribution to higher order, often lower intensity, diffraction spots that contain more orientation information.
- The step size used for the template library need not be as small as possible. A step size of 0.1° does not necessarily mean a template matching angular resolution of 0.1°. A step size of 0.5° - 1° is a good balance of precision and computational consideration given that the overall angular resolution, even in the best case, is 0.5°.
- The template matching results across orientation space are heterogeneous: some patterns are more difficult to match than others, examples being [112], [123] and even the [001], [101] and [111] zones.
- Using simulated data allows for a ground truth and hence the accuracy expressed as pure misorientation. This is a better metric than the pixel-based-intensity cross correlation score.

The conclusions from the experimental tilt series data are:

- Pre-processing has a large effect on the efficacy of template matching, the most important steps being centering, background subtraction and calibration.
- Tilt series can be of great benefit for both identifying mis-indexation and for measuring a sample multiple times to increase the precision and accuracy of the template matching results. In addition, collecting data at different orientation could give a better representation of the sample as shown by $\Sigma 3$ twinning viewed at [112] zone and can reduce overlap between adjacent crystals.
- The conclusions from simulated data on logarithmic scaling and uneven matching results across orientation space are backed up experimentally.

Further more, contributions to Pyxem[12] where made in the form of a automatic calibration function:A.2, and a function for converting the results from template matching into a crystal map from further use in Orix[1]: A.1.

7 Future Work

It has been demonstrated that to collect data at different orientations has added value, although comes at the cost of larger data set sizes and more computation. Therefor, one of the obvious suggested works for the future is more automation to handle tilt series. The tilt series present in this work are essentially 5D data sets, but the work comparing orientations between tilts via for example misorientation to a known or reference, was done mostly manually. It could be possible to cluster the orientations of different grains, recognize the same grain at different tilts and automatically find the misorientation between different tilts. There are challenges in this however. For one; as the sample is tilted, the real space image is distorted, so recognizing grain from tilt to tilt could be difficult to do automatically without good markers that work over the entire tilt range. Another aspect related to tilt series is how to express and visualize accuracy and precision in misorientation space. Here the work by Morawiec [18] should be incorporated into the workflow to better visualize accuracy and precision.

Another problem quite evident and fundamental challenge with template matching is overlapping grains when attempting to find the best template. The patterns produced from this are a combination of multiple different orientations. Suggested solutions include sequential template matching [26] or via Non-Negative Matrix Factorization [3]. Another potential solution to this would be to look through the list of matches for significantly different patterns that could indicate overlap. An alternative solution may be to take the first match and remove the spots present and rematch the pattern. This would require restructuring the data labeling in Pyxem. Currently, in the template library only the positions and intensity of the diffraction spots are stored but they are not indexed. This would also add functionality to Pyxem for plotting these indexed templates onto the patterns that they match to.

All that has been shown and suggested is based on single phase, high symmetry (fcc) examples. Changing one or both of these will certainly introduce new challenges that need solutions. In theory with multi-phase data; if the different phases are significantly different (different lattice parameters and structures), then the fast template matching should be able to handle it. In principle, Cauaerts et. all [5] has demonstrated that this is possible and that increased computational speed is a crucial asset. But it would require addition parameter evaluation as done here for orientation analysis. It should expected that parts of conclusions from this work, for example preprocessing steps, can be applied without too much additional work. However phase that share diffraction spots or have very similar lattice parameters could introduce very difficult problems.

Another factor that was not of much interest with the structures examined in this work is the maximum excitation error used when simulating the templates. For other materials, this parameter could be very sensitive to change [10].

References

[1] Håkon Wiik Ånes et al. *pyxem/orix: orix 0.8.2*. Version v0.8.2. Feb. 2022. DOI: 10.5281/zenodo.6199723. URL: https://doi.org/10.5281/zenodo.6199723.

[2] Jonathan S. Barnard, Duncan N. Johnstone and Paul A. Midgley. 'High-resolution scanning precession electron diffraction: Alignment and spatial resolution'. In: *Ultramicroscopy* 174 (2017), pp. 79–88. ISSN: 0304-3991. DOI: https://doi.org/10.1016/j.ultramic.2016.12.018. URL: https://www.sciencedirect.com/science/article/pii/S030439911630211X.

[3] T. BERGH et al. 'Nanocrystal segmentation in scanning precession electron diffraction data'. In: *Journal of Microscopy* 279.3 (2020), pp. 158–167. DOI: https://doi.org/10.1111/jmi.12850. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jmi.12850. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/jmi.12850.

[4] bsavitzky et al. *py4dstem/py4DSTEM: DOI release*. Version v0.5. July 2019. DOI: 10.5281/zenodo.3333960. URL: https://doi.org/10.5281/zenodo.3333960.

[5] Niels Cautaerts et al. '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'. In: *Ultramicroscopy* (2022), p. 113517. DOI: 10.1016/j.ultramic.2022.113517.

[6] Phillip Crout et al. *pyxem/diffsims: diffsims 0.4.2*. Version v0.4.2. Apr. 2021. DOI: 10.5281/zenodo.4697299. URL: https://doi.org/10.5281/zenodo.4697299.

[7] Olaf Engler and Valerie Randle. *Introduction to Texture Analysis*. CRC Press, Nov. 2009. DOI: 10.1201/9781420063660. URL: https://doi.org/10.1201/9781420063660.

[8] Ardeshir Goshtasby. *Image registration. Principles, tools and methods*. Jan. 2012. ISBN: 978-1-4471-2457-3. DOI: 10.1007/978-1-4471-2458-0.

[9] Mario F. Heinig et al. 'Aminopropylsilatrane Linkers for Easy and Fast Fabrication of High-Quality 10 nm Thick Gold Films on SiO2 Substrates'. In: *ACS Applied Nano Materials* 3.5 (2020), pp. 4418–4427. DOI: 10.1021/acsanm.0c00531.

[10] Endre Jacobsen. 'Scanning Precession Electron Diffraction Template Matching for Automated Phase Mapping of Precipitates in 6xxx Aluminium Alloys'. MA thesis. NTNU, 2020.

[11] Anubhav Jain et al. 'Commentary: The Materials Project: A materials genome approach to accelerating materials innovation'. In: *APL Materials* 1.1 (2013), p. 011002. ISSN: 2166532X. DOI: 10.1063/1.4812323. URL: https://doi.org/10.1063/1.4812323.

[12] Duncan Johnstone et al. *pyxem/pyxem: pyxem 0.14.1*. Version v0.14.1. Apr. 2022. DOI: 10.5281/zenodo.6505200. URL: https://doi.org/10.5281/zenodo.6505200.

[13] Charles Kittel. *Introduction to Solid State Physics*. 8th. New York: John Wiley & Sons, Inc., 2004.

[14] Ian MacLaren et al. 'A Comparison of a Direct Electron Detector and a High-Speed Video Camera for a Scanning Precession Electron Diffraction Phase and Orientation Mapping'. In: *Microscopy and Microanalysis* 26.6 (2020), pp. 1110–1116. DOI: 10.1017/S1431927620024411.

[15] Michael P. Marder. *Condensed Matter Physics*. John Wiley & Sons, Inc., Oct. 2010. DOI: 10.1002/9780470949955. URL: https://doi.org/10.1002/9780470949955.

[16] Ben Martineau et al. 'Unsupervised machine learning applied to scanning precession electron diffraction data'. English. In: *Advanced Structural and Chemical Imaging* (2019). ISSN: 2198-0926. DOI: 10.1186/s40679-019-0063-3.

[17] Paul A. Midgley and Alexander S. Eggeman. 'Precession electron diffraction – a topical review'. In: *IUCrJ* 2.1 (Jan. 2015), pp. 126–136. DOI: 10.1107/S2052252514022283. URL: https://doi.org/10.1107/S2052252514022283.

[18] A. Morawiec et al. 'Orientation precision of TEM-based orientation mapping techniques'. In: *Ultramicroscopy* 136 (2014), pp. 107–118. ISSN: 0304-3991. DOI: https://doi.org/10.1016/j.ultramic.2013.08.008. URL: https://www.sciencedirect.com/science/article/pii/S0304399113002532.

[19] *NanoMEGAS*. URL: www.nanomegas.com.

[20] G. Nolze and R. Hielscher. 'Orientations perfectly colored'. In: *Journal of Applied Crystallography* 49.5 (2016), pp. 1786–1802. DOI: https://doi.org/10.1107/S1600576716012942. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1107/S1600576716012942. URL: https://onlinelibrary.wiley.com/doi/abs/10.1107/S1600576716012942.

[21] Colin Ophus. 'Four-Dimensional Scanning Transmission Electron Microscopy (4D-STEM): From Scanning Nanodiffraction to Ptychography and Beyond'. In: *Microscopy and Microanalysis* 25 (May 2019), pp. 1–20. DOI: 10.1017/S1431927619000497.

[22] Francisco de la Peña et al. *hyperspy/hyperspy: Release v1.6.5*. Version v1.6.5. Oct. 2021. DOI: 10.5281/zenodo.5608741. URL: https://doi.org/10.5281/zenodo.5608741.

[23] E. Rauch and Laurent Dupuy. 'Rapid Diffraction Patterns identification through template matching'. In: *Archives of Metallurgy and Materials* 50 (Jan. 2005), pp. 87–99.

[24] E. Rauch and M. Véron. 'Improving angular resolution of the crystal orientation determined with spot diffraction patterns'. In: *Microscopy and Microanalysis - MICROSC MICROANAL* 16 (July 2010), pp. 770–771. DOI: 10.1017/S1431927610059593.

[25] Gurvinder Singh et al. 'Synthesis of Au nanowires with controlled morphological and structural characteristics'. In: *Applied Surface Science* 311 (2014), pp. 780–788. ISSN: 0169-4332. DOI: https://doi.org/10.1016/j.apsusc.2014.05.162. URL: https://www.sciencedirect.com/science/article/pii/S0169433214012033.

[26] Alexia Valery et al. 'Indexation of diffraction patterns for overlapping crystals in TEM thin foils - Application to orientation mappings'. In: *European Microscopy Congress 2016: Proceedings*. John Wiley & Sons, Ltd, 2016, pp. 657–658. ISBN: 9783527808465. DOI: https://doi.org/10.1002/9783527808465.EMC2016.6091. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9783527808465.EMC2016.6091. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9783527808465.EMC2016.6091.

[27] R. Vincent and P.A. Midgley. 'Double conical beam-rocking system for measurement of integrated electron diffraction intensities'. In: *Ultramicroscopy* 53.3 (1994), pp. 271–282. ISSN: 0304-3991. DOI: https://doi.org/10.1016/0304-3991(94)90039-6. URL: https://www.sciencedirect.com/science/article/pii/0304399194900396.

[28] Matthew Weyland and Paul Midgley. 'Electron Tomography'. In: *Transmission Electron Microscopy: Diffraction, Imaging, and Spectrometry*. Ed. by C. Barry Carter and David B. Williams. Cham: Springer International Publishing, 2016, pp. 343–376. ISBN: 978-3-319-26651-0. DOI: 10.1007/978-3-319-26651-0_12. URL: https://doi.org/10.1007/978-3-319-26651-0_12.

[29] David B. Williams and C. Barry Carter. *Transmission Electron Microscopy*. Springer US, 2009. DOI: 10.1007/978-0-387-76501-3. URL: https://doi.org/10.1007/978-0-387-76501-3.

A Code Contributions to Pyxem

A.1 results_dict_to_crystal_map

Function to take the results from template matching and make them into a crystal map for use in Orix. Start by me and written by myself and Håkon Wiik Ånes.

```
    def results_dict_to_crystal_map(
    results, phase_key_dict, diffraction_library=None, index=None
):
    """Export an indexation result from
    :func:`index_dataset_with_template_rotation` to a crystal map with
    `n_best` rotations, score, mirrors and one phase ID per data point.
    Parameters
    ----------
    results : dict
        Results dictionary obtained from
        :func:`index_dataset_with_template_rotation`.
    phase_key_dict : dict
        Dictionary mapping phase ID to phase name, obtained from
        :func:`index_dataset_with_template_rotation`.
    diffraction_library : diffsims.libraries.DiffractionLibrary, optional
        Used for the structures to be passed to
        :class:`orix.crystal_map.PhaseList`.
    index : int, optional
        Which of the `n_best` solutions (0-indexed) obtained from
        :func:`index_dataset_with_template_rotation` to get a crystal
        map from. Highest allowed value is `n_best` - 1. If not given,
        all solutions are used if `n_best` was more than one and
        `results["phase_index"]` only has one phase, otherwise, only the
        best solution is used.
    Returns
    -------
    orix.crystal_map.CrystalMap
        Crystal map containing `results`. The map has multiple rotations
        and properties ("correlation", "mirrored_template",
        "template_index") per point only if `n_best` passed to
        :func:`index_dataset_with_template_rotation` was more than one
        and "phase_index" only has one phase.
    Notes
    -----
    Phase's :attr:`~orix.crystal_map.Phase.point_group` must be set
    manually to the correct :class:`~orix.quaternion.Symmetry` after
    the crystal map is returned.
    Examples
    --------
    After getting `results` and `phase_key_dict` from template matching
    >>> xmap = results_dict_to_crystal_map(results, phase_key_dict)  # doctest: +SKIP
    >>> xmap.plot()  # Phase map  # doctest: +SKIP
    Getting the second best match if `n_best` passed to the template
    matching function is greater than one
    >>> xmap2 = results_dict_to_crystal_map(
    ...     results, phase_key_dict, index=1
    ... )  # doctest: +SKIP
    """
    ny, nx, n_best = results["phase_index"].shape
    if index is not None and index > n_best - 1:
        raise ValueError(f"`index` cannot be higher than {n_best - 1} (`n_best` - 1)")
```

```
    n_points = nx * ny

    # Phase ID (only one per point is allowed, always)
    if index is None:
        phase_id = results["phase_index"][:, :, 0].ravel()
    else:
        phase_id = results["phase_index"][:, :, index].ravel()
    n_phases = np.unique(phase_id).size

    x, y = np.indices((nx, ny)).reshape((2, n_points))

    if diffraction_library is not None:
        structures = diffraction_library.structures
    else:
        structures = None
    phase_list = PhaseList(names=phase_key_dict, structures=structures)

    euler = np.deg2rad(results["orientation"].reshape((n_points, n_best, 3)))
    if index is None and n_phases > 1:
        euler = euler[:, 0]  # Best match only
    elif index is not None:
        euler = euler[:, index]  # Desired match only
    euler = euler.squeeze()  # Remove singleton dimensions
    rotations = Rotation.from_euler(euler)

    props = {}
    for key in ("correlation", "mirrored_template", "template_index"):
        try:
            prop = results[key]
        except KeyError:
            warnings.warn(f"Property '{key}' was expected but not found in `results`")
            continue

        if index is None and n_phases > 1:
            prop = prop[:, :, 0].ravel()  # Best match only
        elif index is not None:
            prop = prop[:, :, index].ravel()  # Desired match only
        else:
            prop = prop.reshape((n_points, n_best))  # All
        props[key] = prop.squeeze()  # Remove singleton dimensions

    return CrystalMap(
        rotations=rotations,
        phase_id=phase_id,
        x=x,
        y=y,
        phase_list=phase_list,
        prop=props,
    )
```

A.2 calibration_utils.py

Function(s) for automated optimization of calibration via correlation score. Not fully optimized but working and shows the concept well.

```python
# -*- coding: utf-8 -*-
# Copyright 2016-2022 The pyXem developers
#
# This file is part of pyXem.
#
# pyXem is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# pyXem is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with pyXem.  If not, see <http://www.gnu.org/licenses/>.


import numpy as np

from diffsims.generators.library_generator import DiffractionLibraryGenerator
from pyxem.utils.indexation_utils import index_dataset_with_template_rotation


def find_diffraction_calibration(
    patterns,
    calibration_guess,
    library_phases,
    lib_gen,
    size,
    max_excitation_error=0.01,
    **kwargs
):
    """Finds the diffraction calibration for a pattern or set of patterns by maximizing correlation scores.
    Parameters
    ----------
    patterns : hyperspy.signals.Signal2D
        Diffration patterns to be iteratively matched to find maximum correlation scores.
    calibration_guess : float
        Inital value for the diffraction calibration in inverse Angstoms per pixel
    library_phases : diffsims.libraries.StructureLibrary
        Dictionary of structures and associated orientations for which
        electron diffraction is to be simulated.
    lib_gen : diffsims.generators.DiffractionLibraryGenerator
        Computes a library of electron diffraction patterns for specified atomic
        structures and orientations.  Used to create the DiffractionLibrary.
    size : integer
        How many different steps to test for the first two iterations. These steps have a size of 1% of the calibration guess.
    max_excitation_error : float
        Maximum exacitation error.  Default is 0.01.
    kwargs
        Keyword arguments passed to :meth:`index_dataset_with_template_rotation`.

    Returns
    -------
    mean_cal : float
        Mean of calibrations found for each pattern.
```

```
    full_corrlines : numpy.ndarray
        Gives the explicit correlation vs calibration values. Shape:(size*2 + 20, 2 , number of
↪   patterns)
    found_cals : numpy.ndarray
        List of optimal calibration values for each pattern. Shape:(number of patterns)
    """

    images = patterns

    num_patterns = images.data.shape[0]
    found_cals = np.full((num_patterns,), calibration_guess)
    full_corrlines = np.zeros((0, 2, num_patterns))

    stepsize = 0.01 * calibration_guess
    # first set of checks
    corrlines = _calibration_iteration(
        images,
        calibration_guess,
        library_phases,
        lib_gen,
        stepsize,
        size,
        num_patterns,
        max_excitation_error,
        **kwargs
    )
    full_corrlines = np.append(full_corrlines, corrlines, axis=0)

    # refined calibration checks
    calibration_guess = full_corrlines[
        full_corrlines[:, 1, :].argmax(axis=0), 0, 0
    ].mean()
    corrlines = _calibration_iteration(
        images,
        calibration_guess,
        library_phases,
        lib_gen,
        stepsize,
        size,
        num_patterns,
        max_excitation_error,
        **kwargs
    )
    full_corrlines = np.append(full_corrlines, corrlines, axis=0)

    # more refined calibration checks with smaller step
    stepsize = 0.001 * calibration_guess
    size = 20
    calibration_guess = full_corrlines[
        full_corrlines[:, 1, :].argmax(axis=0), 0, 0
    ].mean()

    corrlines = _calibration_iteration(
        images,
        calibration_guess,
        library_phases,
        lib_gen,
        stepsize,
        size,
        num_patterns,
```

```
        max_excitation_error,
        **kwargs
    )
    full_corrlines = np.append(full_corrlines, corrlines, axis=0)
    found_cals = full_corrlines[full_corrlines[:, 1, :].argmax(axis=0), 0, 0]

    mean_cal = found_cals.mean()
    return mean_cal, full_corrlines, found_cals


def _calibration_iteration(
    images,
    calibration_guess,
    library_phases,
    lib_gen,
    stepsize,
    size,
    num_patterns,
    max_excitation_error,
    **kwargs
):
    """For use in find_diffraction_calibration.  Controls the iteration of _create_check_diflib over
    ↪  a set of steps.
    Parameters
    ----------
    images : hyperspy.signals.Signal2D
        Diffration patterns to be iteratively matched to find maximum correlation scores.
    calibration_guess : float
        Inital value for the diffraction calibration in inverse Angstoms per pixel
    library_phases : diffsims.libraries.StructureLibrary
        Dictionary of structures and associated orientations for which
        electron diffraction is to be simulated.
    lib_gen : diffsims.generators.DiffractionLibraryGenerator
        Computes a library of electron diffraction patterns for specified atomic
        structures and orientations.  Used to create the DiffractionLibrary.
    stepsize : float
        Stepsize of iteration.
    size : integer
        How many different steps to test.
    num_patterns : integer
        Number of patterns.
    max_excitation_error : float
        Maximum exacitation error.  Default is 0.01.
    kwargs
        Keyword arguments passed to :meth:`index_dataset_with_template_rotation`.

    Returns
    -------
    corrlines : numpy.ndarray
    """
    corrlines = np.zeros((0, 2, num_patterns))
    temp_line = np.zeros((1, 2, num_patterns))
    cal_guess_greater = calibration_guess
    cal_guess_lower = calibration_guess
    for i in range(size // 2):
        temp_line[0, 0, :] = cal_guess_lower
        temp_line[0, 1, :] = _create_check_diflib(
            images,
            cal_guess_lower,
            library_phases,
```

```
            lib_gen,
            max_excitation_error,
            **kwargs
        )
        corrlines = np.append(corrlines, temp_line, axis=0)

        temp_line[0, 0, :] = cal_guess_greater
        temp_line[0, 1, :] = _create_check_diflib(
            images,
            cal_guess_greater,
            library_phases,
            lib_gen,
            max_excitation_error,
            **kwargs
        )
        corrlines = np.append(corrlines, temp_line, axis=0)

        cal_guess_lower = cal_guess_lower - stepsize
        cal_guess_greater = cal_guess_greater + stepsize

    return corrlines


def _create_check_diflib(
    images, calibration_guess, library_phases, lib_gen, max_excitation_error, **kwargs
):
    """For use in find_diffraction_calibration via _calibration_iteration.  Creates a new
    ↪  DiffractionLibrary from the inputs and then matches it the images.
    Parameters
    ----------
    images : hyperspy.signals.Signal2D
        Diffration patterns to be iteratively matched to find maximum correlation scores.
    calibration_guess : float
        Inital value for the diffraction calibration in inverse Angstoms per pixel
    library_phases : diffsims.libraries.StructureLibrary
        Dictionary of structures and associated orientations for which
        electron diffraction is to be simulated.
    lib_gen : diffsims.generators.DiffractionLibraryGenerator
        Computes a library of electron diffraction patterns for specified atomic
        structures and orientations.  Used to create the DiffractionLibrary.
    max_excitation_error : float
        Maximum exacitation error.  Default is 0.01.
    kwargs
        Keyword arguments passed to :meth:`index_dataset_with_template_rotation`.

    Returns
    -------
    correlations : numpy.ndarray
    """

    half_shape = (images.data.shape[-2] // 2, images.data.shape[-1] // 2)
    reciprocal_r = np.sqrt(half_shape[0] ** 2 + half_shape[1] ** 2) * calibration_guess
    diff_lib = lib_gen.get_diffraction_library(
        library_phases,
        calibration=calibration_guess,
        reciprocal_radius=reciprocal_r,
        half_shape=half_shape,
        with_direct_beam=False,
        max_excitation_error=max_excitation_error,
    )
```

```
    result, phasedict = index_dataset_with_template_rotation(images, diff_lib, **kwargs)
    correlations = result["correlation"][:, :, 0].flatten()
    return correlations
```

B Crystallographic Information Files used

B.1 Si

```
# generated using pymatgen
data_Si
_symmetry_space_group_name_H-M   'P 1'
_cell_length_a   5.46872800
_cell_length_b   5.46872800
_cell_length_c   5.46872800
_cell_angle_alpha   90.00000000
_cell_angle_beta   90.00000000
_cell_angle_gamma   90.00000000
_symmetry_Int_Tables_number   1
_chemical_formula_structural   Si
_chemical_formula_sum   Si8
_cell_volume   163.55317139
_cell_formula_units_Z   8
loop_
 _symmetry_equiv_pos_site_id
 _symmetry_equiv_pos_as_xyz
  1  'x, y, z'
loop_
 _atom_site_type_symbol
 _atom_site_label
 _atom_site_symmetry_multiplicity
 _atom_site_fract_x
 _atom_site_fract_y
 _atom_site_fract_z
 _atom_site_occupancy
  Si  Si0  1  0.25000000  0.75000000  0.25000000  1
  Si  Si1  1  0.00000000  0.00000000  0.50000000  1
  Si  Si2  1  0.25000000  0.25000000  0.75000000  1
  Si  Si3  1  0.00000000  0.50000000  0.00000000  1
  Si  Si4  1  0.75000000  0.75000000  0.75000000  1
  Si  Si5  1  0.50000000  0.00000000  0.00000000  1
  Si  Si6  1  0.75000000  0.25000000  0.25000000  1
  Si  Si7  1  0.50000000  0.50000000  0.50000000  1
```

B.2 Au

```
# generated using pymatgen
data_Au
```

```
_symmetry_space_group_name_H-M   'P 1'
_cell_length_a   4.17128800
_cell_length_b   4.17128800
_cell_length_c   4.17128800
_cell_angle_alpha   90.00000000
_cell_angle_beta   90.00000000
_cell_angle_gamma   90.00000000
_symmetry_Int_Tables_number   1
_chemical_formula_structural   Au
_chemical_formula_sum   Au4
_cell_volume   72.57892447
_cell_formula_units_Z   4
loop_
 _symmetry_equiv_pos_site_id
 _symmetry_equiv_pos_as_xyz
  1  'x, y, z'
loop_
 _atom_site_type_symbol
 _atom_site_label
 _atom_site_symmetry_multiplicity
 _atom_site_fract_x
 _atom_site_fract_y
 _atom_site_fract_z
 _atom_site_occupancy
  Au  Au0  1  0.00000000  0.00000000  0.00000000  1
  Au  Au1  1  0.00000000  0.50000000  0.50000000  1
  Au  Au2  1  0.50000000  0.00000000  0.50000000  1
  Au  Au3  1  0.50000000  0.50000000  0.00000000  1
```

B.3 Cu

```
# generated using pymatgen
data_Cu
_symmetry_space_group_name_H-M   'P 1'
_cell_length_a   3.62126200
_cell_length_b   3.62126200
_cell_length_c   3.62126200
_cell_angle_alpha   90.00000000
_cell_angle_beta   90.00000000
_cell_angle_gamma   90.00000000
_symmetry_Int_Tables_number   1
_chemical_formula_structural   Cu
_chemical_formula_sum   Cu4
_cell_volume   47.48755856
_cell_formula_units_Z   4
loop_
 _symmetry_equiv_pos_site_id
 _symmetry_equiv_pos_as_xyz
  1  'x, y, z'
loop_
 _atom_site_type_symbol
 _atom_site_label
 _atom_site_symmetry_multiplicity
 _atom_site_fract_x
 _atom_site_fract_y
```

```
 _atom_site_fract_z
 _atom_site_occupancy
  Cu  Cu0  1  0.00000000  0.00000000  0.00000000  1
  Cu  Cu1  1  0.00000000  0.50000000  0.50000000  1
  Cu  Cu2  1  0.50000000  0.00000000  0.50000000  1
  Cu  Cu3  1  0.50000000  0.50000000  0.00000000  1
```

C General Template Matching Notebook

Example code that from loading the data set, to template matching and finally inspection of the results. Available as a notebook along with less general workflows on github:

https://github.com/soupmongoose/Pyxem-Notebooks

```
%matplotlib qt

import numpy as np
import matplotlib.pyplot as plt
import hyperspy.api as hs

# Loading Data set and basic inspection
dpau = hs.load("Data/20210714 142621 SPED B/128x128_12cm_1deg_step20_1nm_NBDalpha5_TX5.hspy" )
dpau

dpau.plot(vmax = 50, cmap = 'viridis')
dpau = dpau.inav[35:110,15:110]
dpau
# Centering, a required step
dpau.center_direct_beam(method="blur", half_square_width=10, sigma=3)
dpau.plot(vmax = 50, cmap = 'viridis')
# Background subtraction, required for using logarithmic scaling
from skimage import filters
# Set values lower than a specific value to 0 in the image
def crop_minimum(image, minimum=0.0005):
    copied = image.copy()
    copied[copied <= minimum] = 0.
    return copied

# difference of gaussians serves to remove the background intensity
temp = dpau.inav[93,85].subtract_diffraction_background(method="difference of gaussians",
                                                        min_sigma=9,
                                                        max_sigma=10, )
# smooth out the output
temp = temp.map(filters.gaussian, sigma=0.5, inplace=False)
temp = temp.map(crop_minimum, minimum = 1, inplace=False)
temp = temp.map(filters.gaussian, sigma=0.5, inplace=False)

temp = dpau.inav[93,85]
temp.plot(scalebar = False, cmap = 'inferno_r', norm = 'symlog')

dpau = dpau.subtract_diffraction_background(method="difference of gaussians",
                                            min_sigma=9,
                                            max_sigma=10,
                                           lazy_result = False)
dpau.map(filters.gaussian, sigma=0.5)
dpau.map(crop_minimum, minimum = 1)
```

```
dpau.map(filters.gaussian, sigma=0.5)
dpau.plot(vmax = 50, cmap = 'viridis')

# Creating the Template Library
from diffsims.generators.rotation_list_generators import get_beam_directions_grid
resolution = 1
grid_cub = get_beam_directions_grid("cubic", resolution, mesh="spherified_cube_edge")
print(grid_cub.shape[0])

# this cell serves to visualize the grid of orientations in stereographic projection
from orix.quaternion import Orientation, symmetry
from orix.vector import Vector3d
#create orientations from the grid
origrid = Orientation.from_euler(
    np.radians(grid_cub),
    symmetry=symmetry.Oh
)

v = Vector3d(((0, 0, 1))) #looking in the Z direction
origrid.scatter("ipf", direction=v, s = 10)

import diffpy
from diffsims.libraries.structure_library import StructureLibrary
from diffsims.generators.diffraction_generator import DiffractionGenerator
from diffsims.generators.library_generator import DiffractionLibraryGenerator

# Parameters necessary for simulating a template library
diffraction_calibration = 0.00938
half_shape = (dpau.data.shape[-2]//2, dpau.data.shape[-1]//2)
reciprocal_radius = np.sqrt(half_shape[0]**2 + half_shape[1]**2)*diffraction_calibration

structure_matrix = diffpy.structure.loadStructure("Data/Au_mp-81_conventional_standard.cif")

# In essence the microscope setup
diff_gen = DiffractionGenerator(accelerating_voltage=200,
                                precession_angle=1,
                                scattering_params=None,
                                shape_factor_model="linear",
                                minimum_intensity=0.1,
                                )

lib_gen = DiffractionLibraryGenerator(diff_gen)

library_phases_Au = StructureLibrary(["Au"], [structure_matrix], [grid_cub])

diff_lib_Au = lib_gen.get_diffraction_library(library_phases_Au,
                                              calibration=diffraction_calibration,
                                              reciprocal_radius=reciprocal_radius,
                                              half_shape=half_shape,
                                              with_direct_beam=False,
                                              max_excitation_error=0.08)

from pyxem.utils import indexation_utils as iutls
from pyxem.utils import plotting_utils as putls

# shows how the in-plane angle is optimized for a single pattern and single template
image = dpau.inav[93,85].data
simulation_test = diff_lib_Au["Au"]["simulations"][92]

a, c = iutls.get_in_plane_rotation_correlation(
```

```
    image,
    simulation_test,
    intensity_transform_function=None,  n
    delta_r = 1,
    delta_theta = 0.1,
    max_r = None,
    find_direct_beam = False,
    direct_beam_position = None,
    normalize_image=True,
    normalize_template=True,
)

fig, ax = plt.subplots()
ax.plot(a, c)
ax.set_xlim(0, 360)
ax.set_xlabel("Angular shift (Degrees)")
ax.set_ylabel("Correlation")


putls.plot_template_over_pattern(image,
                                 simulation_test,
                                 in_plane_angle=a[np.argmax(c)],
                                 coordinate_system = "cartesian",
                                 size_factor = 10,
                                 vmax=20,
                                 max_r = 200,
                                 find_direct_beam=True,
                                 cmap = "inferno"
                                )

# Matching a single pattern

simulations = diff_lib_Au["Au"]["simulations"]

delta_r = 1
delta_theta = 1
max_r = None
intensity_transform_function = None
find_direct_beam = False
direct_beam_position = None
normalize_image = True
normalize_templates = True

n_best = 10
indices_n, angles_n, correlations_n, signs_n = iutls.get_n_best_matches(image,
                                               simulations,
                                               n_best,
                                               frac_keep,
                                               n_keep,
                                               delta_r,
                                               delta_theta,
                                               max_r,
                                               intensity_transform_function,
                                               find_direct_beam,
                                               direct_beam_position,
                                               normalize_image,
                                               normalize_templates,
                                              )
```

```
mirrored = signs_n[0] == -1
putls.plot_template_over_pattern(image,
                                 simulations[indices_n[0]],
                                 in_plane_angle=angles_n[0],
                                 coordinate_system = "cartesian",
                                 size_factor = 60,
                                 max_r = 200,
                                 vmax = 20,
                                 mirrored_template=mirrored,
                                 find_direct_beam=False,
                                 cmap = "inferno_r",
                                 marker_color = 'darkgreen',
                                 marker_type = 'x',
                                )


# Main template matching function

def log_func(x):
    return(np.log10(x + 0.01)) # log function for better matching results

intensity_transform_function = log_func

result, phasedict = iutls.index_dataset_with_template_rotation(dpau,
                                                               diff_lib_Au,
                                                               phases = ["Au"],
                                                               n_best = n_best,
                                                               frac_keep = frac_keep,
                                                               n_keep = n_keep,
                                                               delta_r = delta_r,
                                                               delta_theta = delta_theta,
                                                               max_r = None,
                                                               intensity_transform_function=log_func,
                                                               normalize_images = normalize_image,
                                                               normalize_templates=normalize_templates,
                                                               target= 'cpu'
                                                               )

# Converting Results to crystal map for use with Orix
xmap = iutls.results_dict_to_crystal_map(result,phasedict)

xmap.phases[0].space_group = 225
xmap

# Verification of matching results

from orix import plot
from orix.crystal_map import CrystalMap, Phase, PhaseList
from orix.io import load, save
from orix.quaternion import Orientation, Rotation, symmetry
from orix.vector import Vector3d

ckey_m3m = plot.IPFColorKeyTSL(xmap.phases["Au"].point_group, direction=Vector3d.xvector())
rgb_Aux = ckey_m3m.orientation2color(xmap["Au"].orientations)
ckey_m3m.direction = Vector3d.yvector()
rgb_Auy = ckey_m3m.orientation2color(xmap["Au"].orientations)
ckey_m3m.direction = Vector3d.zvector()
rgb_Auz = ckey_m3m.orientation2color(xmap["Au"].orientations)


```

```
#plotting the orientation map from all three principle directions
fig = plt.figure()
ax0 = fig.add_subplot(131, projection="plot_map")
ax1 = fig.add_subplot(132, projection="plot_map")
ax2 = fig.add_subplot(133, projection="plot_map")
ax0.set_title("X")
ax1.set_title("Y")
ax2.set_title("Z")
ax0.plot_map(xmap, rgb_Aux)
ax1.plot_map(xmap, rgb_Auy)
ax2.plot_map(xmap, rgb_Auz)


#plot correlation score
xmap.plot(value = xmap.correlation[:,0])

xmap.phases.add_not_indexed() # not indexed for the substrate/vacuum
xmap.phases
xmap[(xmap.correlation[:,0] < 0.0005)].phase_id = -1
xmap

xmap.plot()

rgb_Auz = ckey_m3m.orientation2color(xmap["Au"].orientations)
rgb_all = np.zeros((xmap.size, 3))
rgb_all[xmap.phase_id == 0] = rgb_Auz
rgb_all[xmap.phase_id == -1] = [0,0,0]
xmap.plot(rgb_all)

solution = result["orientation"]
#scan coordinate to check
px = 72
py = 65
# which solution to plot
n_sol = 0

index = np.ravel_multi_index((py,px),xmap.shape)
# query the necessary info from the solution
sim_sol_index = xmap.template_index[index,0]
mirrored_sol = xmap.mirrored_template[index,0]
in_plane_angle = np.rad2deg(xmap.rotations[index,0].to_euler())[0][0]
# # query the appropriate template
sim_sol = simulations[sim_sol_index]

fig = plt.figure()
ax0 = fig.add_subplot(121, projection="plot_map")
ax1 = fig.add_subplot(122)
ax0.set_title("Z")
ax0.plot_map(xmap, rgb_all)
ax0.scatter(px,py, marker = 'X', c = 'g', s = 50)
putls.plot_template_over_pattern(dpau.inav[px, py].data,
                                 sim_sol,
                                 ax = ax1,
                                 in_plane_angle=in_plane_angle,
                                 coordinate_system = "cartesian",
                                 size_factor = 10,
                                 vmax=20,
                                 max_r = 200,
                                 mirrored_template=mirrored_sol,
                                 find_direct_beam=True,
```

```
                                cmap = "inferno",
                                marker_color = "green"
                               )


# Creating a plotting normalized misorientation angle map
orifirst = xmap['Au'].orientations
store = np.zeros((xmap['Au'].size,5))
for i in range (5):
    corr_weight = xmap['Au'].correlation[:,i]/xmap['Au'].correlation[:,0]
    orinext = Orientation(xmap['Au'].rotations[:,i],symmetry=symmetry.Oh)
    store[:,i] = np.degrees((orinext - orifirst).angle.data) * corr_weight
mean_misori_weighted = np.mean(store[:,1::], axis = 1)

xmap['Au'].plot(value = mean_misori_weighted,scalebar = False, vmax = 10, colorbar = True)
```

D Simulated data code

Important functions for creating random simulated data sets and rotating them. Full notebooks on github: https://github.com/soupmongoose/Pyxem-Notebooks

```
%matplotlib qt

import numpy as np
import pyxem as pxm
import hyperspy.api as hs
import matplotlib.pyplot as plt
import math
import random
import diffpy
from diffsims.libraries.structure_library import StructureLibrary
from diffsims.generators.diffraction_generator import DiffractionGenerator
from diffsims.generators.library_generator import DiffractionLibraryGenerator
from pyxem.utils import indexation_utils as iutls
from pyxem.utils import plotting_utils as putls
from pyxem.utils import polar_transform_utils as ptutls

from diffsims.generators.rotation_list_generators import get_beam_directions_grid
grid_cub = get_beam_directions_grid("cubic", 0.1, mesh="spherified_cube_edge")
print("Number of patterns: ", grid_cub.shape[0])

# Parameters necessary for simulating a template library
diffraction_calibration = 0.01
half_shape = (256//2, 256//2)
reciprocal_radius = np.sqrt(half_shape[0]**2 + half_shape[1]**2)*diffraction_calibration

# importing the structures
structure_matrix = diffpy.structure.loadStructure("Au_mp-81_conventional_standard.cif")


diff_gen = DiffractionGenerator(accelerating_voltage=200,
                                precession_angle=1,
                                scattering_params=None,
                                shape_factor_model="linear",
                                minimum_intensity=0.1,
```

```
                                )

lib_gen = DiffractionLibraryGenerator(diff_gen)

library_phases = StructureLibrary(["phase"], [structure_matrix], [grid_cub])

diff_lib_for_data = lib_gen.get_diffraction_library(library_phases,
                                        calibration=diffraction_calibration,
                                        reciprocal_radius=reciprocal_radius,
                                        half_shape=half_shape,
                                        with_direct_beam=False,
                                        max_excitation_error=0.08)

#Function for creating a simulated data of ALL templates present in simulations with a set inplane
def full_lib_thing(simulations,inplane):
    allpatterns = hs.signals.Signal2D(np.zeros((1,simulations.size,256,256)))
    allpatterns.set_signal_type(signal_type="electron_diffraction")
    pattern = hs.signals.Signal2D(np.zeros((256,256)))
    pattern.set_signal_type(signal_type="electron_diffraction")

    for i in range(simulations.size):
        pattern = pointplotter(simulations,inplane,i,simulations.size)
        allpatterns.inav[i,:] = pattern
        pattern = hs.signals.Signal2D(np.zeros((256,256)))
    return allpatterns

#*Code to create a random data set of simulated data and return the eulers and indexes used

def grab_random_pattern(simulations):
    inplane = random.randint(0,360)
    template = random.randint(0,simulations.size-1)
    pattern = pointplotter(simulations,inplane,template,5)
    return pattern, inplane, template
def build_random_tilts(diff_lib,n):
    orientations = diff_lib['phase']['orientations']
    simulations = diff_lib['phase']['simulations']
    pattern_set = hs.signals.Signal2D(np.zeros((1,n,256,256)))
    pattern_set.set_signal_type(signal_type="electron_diffraction")
    eulers = np.zeros((n,3))
    template_indexs = np.zeros((1,n))
    for i in range(n):
        pattern, inplane, template = grab_random_pattern(simulations)
        euler = orientations[template].copy()
        euler[0] = euler[0] + inplane
        pattern_set.inav[i,0] = pattern
        eulers[i,:] = euler
        template_indexs[0,i] = template
    return pattern_set,eulers,template_indexs

from pyxem.utils.plotting_utils import get_template_cartesian_coordinates
from diffsims.pattern.detector_functions import add_shot_and_point_spread

#*function for taking the simulated template and making a pattern
def pointplotter(simulations,inplane,template,sigma):
    pattern = hs.signals.Signal2D(np.zeros((256,256)))
    pattern.set_signal_type(signal_type="electron_diffraction")
    x, y, intensities = get_template_cartesian_coordinates(simulations[template],in_plane_angle=inplane,window_size=(255
    for i in range(x.size):
        pattern.isig[x[i],y[i]]=intensities[i]
    pattern = add_shot_and_point_spread(pattern.T, sigma, shot_noise=False)
```

```
    pattern = hs.signals.Signal2D(pattern)
    return pattern

randompatterns, eulers, template_indexs = build_random_tilts(diff_lib_for_data,4000)

# Plotting function to show a movement of tilts through a z-ipf
def plot_matchedandtrue_Oh(num):
    oritrue = Orientation.from_euler(
        np.radians(eulers[:,:,:]),
        symmetry=symmetry.Oh
    )
    orimatched = Orientation.from_euler(
    np.radians(result['orientation'][num,:,0,:]),
    symmetry=symmetry.Oh
    )
    cmap = np.linspace(0, 1, oritrue.size)
    v = Vector3d((0, 0, 1))
    oritrue.scatter("ipf", direction=v, c=cmap)
    orimatched.scatter("ipf", direction=v, c=cmap)

randompatterns.plot(cmap = 'viridis')

#*Function to rotated the simulated patterns
def rotate_sims(random_tilts,eulers,rot_quat,num_rot,pg):
    shape = random_tilts.data.shape
    pattern_set = hs.signals.Signal2D(np.zeros((shape[0],shape[1] + num_rot,shape[2],shape[3])))
    pattern_set.inav[0,:] = random_tilts.data[:,0,:,:]
    all_eulers = np.zeros((shape[0],num_rot + 1,3))
    all_eulers[:,0,:] = eulers
    for i in range(0, num_rot):
#         rotated_eulers = np.degrees(om2eu(np.matmul(eu2om(np.radians(all_eulers[:,i,:])) ,rotation)))
        quats = Orientation.from_euler(np.radians(all_eulers[:,i,:]),symmetry=pg)
        rotated_quats = (quats * rot_quat)
        rotated_quats.symmetry = pg
        rotated_eulers = np.degrees(rotated_quats.to_euler())

        all_eulers[:,i+1,:] = rotated_eulers
        diff_gen = DiffractionGenerator(accelerating_voltage=200,
                                    precession_angle=1,
                                    scattering_params=None,
                                    shape_factor_model="linear",
                                    minimum_intensity=0.1,
                                    )

        lib_gen = DiffractionLibraryGenerator(diff_gen)
        library_phases = StructureLibrary(["phase"], [structure_matrix], [rotated_eulers])
        diff_lib = lib_gen.get_diffraction_library(library_phases,
                                               calibration=diffraction_calibration,
                                               reciprocal_radius=reciprocal_radius,
                                               half_shape=half_shape,
                                               with_direct_beam=False,
                                               max_excitation_error=0.08)
        for j in range(eulers.shape[0]):
            pattern = pointplotter(diff_lib['phase']['simulations'],0,j,5)
            pattern_set.inav[i+1,j] = pattern

    return(pattern_set,all_eulers)
```