NTNU
Norwegian University of
Science and Technology

Atomic Resolution 3-D Reconstruction of Crystal Grain Boundaries

Elise Otterlei Brenne

Master of Science in Physics and Mathematics
Submission date: May 2018
Supervisor: Randi Holmestad, IFY
Co-supervisor: Yuichi Ikuhara, University of Tokyo
Bin Feng, University of Tokyo

Norwegian University of Science and Technology
Department of Physics

Preface

This thesis is submitted as the conclusion of a five year integrated master program in Applied Physics and Mathematics at the Norwegian University of Science and Technology (NTNU Trondheim). The work presented here was carried out during my stay as an exchange student at University of Tokyo, October 2017 to April 2018.

My work this semester was mainly focused on developing a method for 3-D reconstruction of the crystal structure of a crystal grain boundary in atomic resolution. The experimental images used in the reconstruction are acquired by Bin Feng and Ryo Ishikawa in the facilities of the Crystal Interface laboratory at the University of Tokyo. The idea to the project was developed during my summer internship in this laboratory, where I learned about different electron microcope imaging techniques and some theory of electron tomography. One aim of this project is that the developed reconstruction method should be easily accessible for people interested in applying tomography for grain boundary research. Some work was therefore done on making a tutorial on how to use the code developed for both image preparation and reconstruction. Due to limited time, the tutorial only covers the first part of the reconstruction process concerning the preparation of the experimental images.

I would like to thank my supervisor Prof. Yuichi Ikuhara for giving me a warm welcome at the Crystal Interface Laboratory. I am grateful to him, his co-workers and the students in the lab for making my stay in Tokyo an invaluable and enjoyable experience. Furthermore, I would like to thank co-supervisor Asst. Prof. Bin Feng for introducing me to the fascinating topic of electron tomography, and for excellent guidance. Also, I would like to thank my supervisor RandiHolmestad at the Department of Physics, NTNU, and Asst. Prof. Ryo Ishikawa for valuable help and discussions throughout the work of this thesis.

Tokyo, May 2, 2018

Elise Otterlei Brenne

Abstract

Grain boundaries are typical crystal defects, which to a large extent determines the macroscopic properties of the material. In order to control these properties and design new materials, detailed knowledge about the atomic structure of such defects is essential. As atomic resolution has become routinely available in scanning transmission electron microscopy (STEM), tools for obtaining three-dimensional (3-D) information from the projection data are valuable.

In this thesis, a method for 3-D reconstruction of the atomic structure in crystal grain boundaries is developed. The method is based on tomographic reconstruction from images acquired by high-angle annular dark field (HAADF) STEM. Through application on a model grain boundary in yttrium-stabilized zirconia (YSZ), an important material in energy applications, the method was demonstrated to accurately determine the 3-D positions of all atoms in the structure from only three projection images. This revealed information about the grain boundary structure that has not previously been available, providing details that could not be identified in the projected images alone. A precision of 4 pm was reached, estimated from comparison with the well-known bulk structure of YSZ. Moreover, as the developed reconstruction method enables direct 3-D structure retrieval from experimental images, it provides an accurate atomic scale model which can serve as an ideal starting point for further theoretical investigations.

Methods for preparing the experimental images are also developed, including distortion correction, noise reduction and atom column localization, crucial for an accurate reconstruction. These, together with the tomographic reconstruction method, are implemented in Python. A tutorial is written to illustrate how to use the code, included in the appendices of this thesis, contributing to making tomography an easily accessible tool in GB research.

Samandrag

Korngrenser er typiske defekter i krystallar som i stor grad bestemmer dei fysiske eigenskapane materialet har. For å kunne kontrollere desse eigenskapane og utvikle nye materialar er kunnskap om strukturen i slike defekter svært viktig. Med atom-nivå oppløysing tilgjengeleg i skanning transmisjonselektronmikroskopi (STEM) er verkty som gir moglegheit til å få tre-dimensjonal informasjon frå projeksjonsdataane verdifulle.

I denne masteroppgåva har ei metode for 3-D rekonstruering av atomstrukturen i krystallkorngrenser blitt utvikla. Metoden er basert på tomografisk rekonstruksjon frå bilder tatt med høgvinkel annular darkfield STEM (HAADF-STEM). Gjennom å rekonstruere ein modell av ei korngrense i yttria-stabilisert zirkonia (YSZ), eit viktig materiale med bruksområder blant anna innan energi, blir det demonstrert at metoden kan bestemme posisjonane til alle atoma frå så få som tre bilder. Dette gav ny informasjon om strukturen i korngrensa som ikkje var mogleg å sjå i dei 2-D projeksjonsbilda åleine. Presisjonen den rekonstruerte strukturen er målt til 4 pm, estimert ved å samanlikne strukturen med den teoretiske strukturen til ein enkeltkrystall av YSZ. Vidare, sidan metoden gjer det mogleg å trekke 3-D informasjon direkte utifrå eksperimentelle bilder, gir den ein atom-skala modell som er eit godt utgangspunkt for vidare teoretisk utforsking.

Metoder er også utvikla for å behandle dei eksperimentelle bilda, med korreksjon av feil frå forstyrringar under bildetakinga, støyredusering og lokalisering av atom-kolonner, som er viktige steg for å få til ein nøyaktig rekonstruksjon. Alle metodane er implementerte i Python og ein skildring av korleis bruke koden er inkludert i appendix, noko som kan bidra til å gjere tomografi til eit lett tilgjengeleg verkty i korngrenseforsking.

List of Abbreviations

ADF	Annular dark-field
AET	Atomic electron tomography
ART	Algebraic reconstruction technique
BCC	Body-centered cubic
BF	Bright-field
CCF	Cross-correlation function
CM	Center-of-mass
CSL	Coincidence site lattice
CT	Computed tomography
DBP	Direct back-projection
DFT	Density functional theory
DT	Discrete tomography
EDX	Energy-dispersive X-ray spectroscopy
EELS	Electron energy loss spectroscopy
FBP	Filtered back-projection
FCC	Face-centered cubic
GB	Grain boundary
GBS	Grain boundary segregation
HAADF	High-angle annular dark field
HAGB	High-angle grain boundary
HRTEM	High-resolution transmission electron microscopy
LAGB	Low-angle grain boundary
MC	Monte Carlo
MD	Molecular Dynamics
RMSD	Root-mean-square deviation
SART	Simultaneous algebraic reconstruction technique
SC	Simple cubic
SIRT	Simultaneous iterative reconstruction technique
SNR	Signal-to-noise ratio
SOCs	Solid oxide cells
STEM	Scanning transmission electron microscope
TDS	Thermal diffuse scattering
TEM	Transmission electron microscope
YSZ	Yttrium-stabilized zirconia
2-D	Two-dimensional
3-D	Three-dimensional

Contents

Chapter 1

Introduction

1.1 Motivation

In nature, perfect crystals are rare. Real materials contain defects like dislocations, interfaces and point defects, having important ramifications for the macroscopic properties of the material[1]. One common type of crystal defects are grain boundaries, defined as the interface between adjacent single crystals, or grains. With an atomic structure and chemical composition often very different from that of the single crystal, grain boundaries have exhibited unusual properties, both mechanically[2], chemically[3] and electrically[4, 5], having great impact on the behaviour of many technologically important materials. In materials engineering, one seeks to tailor the type of grain boundaries occurring in a material, enabling design and development of high-performance materials. To achieve this, fundamental knowledge about grain boundary effects is essential, which is the reason why this has been a major area of focus for researchers during the past century[6].

The transmission electron microscope (TEM) is a powerful tool in materials characterization, contributing to countless discoveries since its invention by Max Knoll and Ernst Ruska in 1931[7]. In particular, high-angle annular dark field scanning TEM (HAADF-STEM) has become popular due to its intuitively interpretable images[8]. With major improvents in resolution in recent years, sub-Ångström resolution has been reached[9],

enabling imaging of individual atom column in crystals, as illustrated in Figure 1.1. The atomic structure in some types of grain boundaries can also be imaged through fabrication of a bicrystal, where the relative orientation of two grains is controlled to design a particular GB model with geometry suitable for STEM imaging.

Inevitably, as HAADF-STEM images are 2-D projections of a 3-D structure, much of the structural information is lost. However, by combining projections from several directions, it is possible to retrieve the information and reconstruct the 3-D structure. This is the challenge in tomography, which has found widespread applications in medicine, physical sciences, biology and engineering. In material science, atomic electron tomography (AET) has been used for 3-D determination of the atomic structure in materials, with successful application to crystals, and defects like stacking faults, dislocations, chemical order/disorder and to determine the shape and chemical structure of nanoparticles[10, 11, 12, 13, 14, 15]. Similarly to conventional tomography, the resolution of AET is limited by the number of projection images, and hence, the technique is most useful when imaging from numerous directions is possible. This is not the case when studying high-angle grain boundaries in bicrystals, as will be discussed in more detail in the following chapters. A new method for 3-D reconstruction is therefore needed.

The atomic structure retrieved by tomographic reconstruction can further be used as input to theoretical calculations. Calculations from *first principles* refers to starting from the very fundamental atomic interactions, with density functional theory (DFT)[16] being one common method. Computer simulations based on e.g. Monte Carlo (MC) methods or molecular dynacims (MD)[17] are another approach to investigating material properties through simulation of the behaviour of the constituents of the material. However, such theoretical calculations rely heavily on average atomic models extracted from crystallography[15], limiting the performance when defects are present. An experiment-based initial 3-D model of the material would be valuable, serving as a suitable starting point for providing more realistic calculation results, going beyond the average crystallographic models.

Yttrium-stabilized zirconia (YSZ) is an important ceramic material with applications in devices for energy storage and conversion. YSZ is commonly used as an electrolyte in e.g. solid oxide fuel cells and solid-state batteries due to its high ionic conductivity and low electron conductivity which is essential in an electrolyte material, in combination with high thermal and chemical stability[18, 19]. The material has therefore been subjected to extensive studies through the last decades, concluding that grain boundaries within the material play a significant role, strongly affecting the overall performance of the device. In particular, ionic transport across grain boundaries is reported to be significantly reduced compared to in the grain interior, an effect attributed to the segregation of dopant yttrium atoms to the GB region[20, 21]. This phenomenon has gained much interest, with studies performed to investigate the driving forces behind the segregation through atomistic simulations[22].

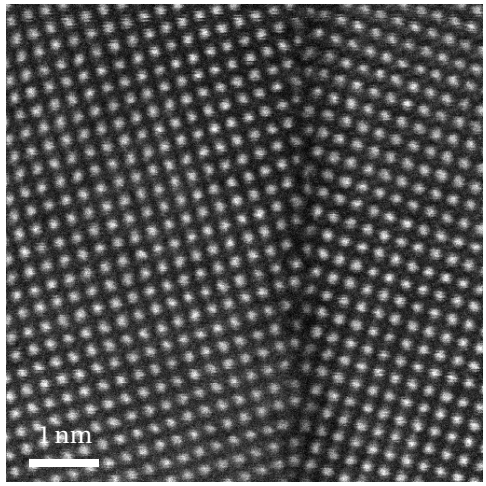


Figure 1.1: Atomic resolution HAADF-STEM image of a Σ5{310}/[001] grain boundary in YSZ along the [001] zone axis. Bright dots are columns of Y and Zr atoms. Image obtained by Bin Feng at the Crystal Interface Laboratory, University of Tokyo.

Obtaining detailed knowledge of the atomic structure in the GB region is seen as key to understand and control the properties of a material[23]. The main focus throughout this thesis is therefore been to establish a method for high-accuracy determination of the atomic structure in grain boundaries. This will be done through combining the atomic resolution available in HAADF-STEM imaging with principles from tomography.

The main objectives are:

1. Development of a method for 3-D tomographic reconstruction of crystal grain boundaries from HAAADF-STEM images at atomic resolution, including establishing techniques for preparation of the experimental images

2. Determination of the atomic structure of a $\Sigma 5\{310\}/[001]$ grain boundary model in YSZ through application of the developed reconstruction method

3. Write code covering the complete process of grain boundary reconstruction, including image preparation, making the developed the method easily accessible to others.

The Crystal Interface Laboratory, University of Tokyo, is conducting fundamental research on property generating mechanisms in materials, with one area of focus being the effects from interfaces and defects within ceramics.The work presented in this thesis is the first attempt to use tomography for 3-D structural characterization in this research environment. Therefore, the focus throughout the thesis will be on establishing a general framework for applying this technique in grain boundary research. This work should provide a future reference on tomographic reconstruction from atomic resolution images at the Crystal Interface Laboratory.

1.2 Structure of the Report

The outline of the rest of the thesis is as follows. Chapter 2 provides theory and background on some important topics for understanding the motivation for the work in this thesis, starting with an introduction to crystals and grain boundaries and their implications for macroscopic material properties. The fundamental principles of STEM imaging are introduced, with focus on HAADF-STEM and EDX, two techniques that are employed to obtained the experimental data utilized in this work. In particular, imaging of crystal lattices and defects is discussed. Thereafter, the concept of tomographic reconstruction is explained in order to provide the reader with the knowledge needed to follow the main steps of the reconstruction method that is developed. Details of this method are provided in Chapter 4, with a step-by-step illustration through

the application on a YZS grain boundary model. In Chapter 5, the final results from the reconstruction of the grain boundary model is presented, along with an evaluation of the developed method, commenting on accuracy of the results, current limitations and possible improvements. Finally, concluding remarks are made in Chapter 6, summarizing the report and highlighting the most important results, before suggestions are given for further work on the topic of the thesis in Chapter 7.

Chapter 2

Theory and Background

This chapter is divided into four parts, introducing some central topics of this thesis. Firstly, a brief introduction to crystallography is given, as the nature of crystal defects are best explained with respect to the perfect crystal structure. Terminology that will be useful throughout the thesis is defined. Thereafter, the concept of grain boundaries is explained and their implications for macroscopic material properties are discussed. The transmission electron microscopy (TEM) and some techniques for material characterization is described, with emphasis on high-angle annular dark field scanning TEM. In combination with tomography, this technique can provide atomic resolution structural information in three dimensions, as will be discussed in the final section. Current limitations are outlined, demonstrating the need for a new method for 3-D reconstruction of grain boundaries.

2.1 Crystallography

Much of the following text is based on elements from the work of Kelly in Crystallography and Crystal Defects[24] and Kittel in Introduction to Solid State Physics[25]. For more comprehensive discussion on the topics introduced in the following, the reader is referred to these texts.

A crystal can be described as a set of atoms (or ions) arranged in a perfectly ordered

periodic structure consisting of several smaller identical units. The *primitive unit cell* is the smallest possible unit which can be repeated to cover the whole structure without overlapping. Each of these units can be related to exactly one point in the *Bravais lattice* of the crystal. The Bravais lattice is spanned by the following vector,

$$\vec{R}_{hkl} = h\vec{a}_1 + k\vec{a}_2 + l\vec{a}_3, \tag{2.1}$$

meaning that with the origin placed in one point of the lattice, all other points can be reached by this vector where h, k and l are integers and $\vec{a}_i$ are the *primitive* vectors. The lengths $|\vec{a}_i|$ are often referred to as the *lattice parameters*. This way, the Bravais lattice contains all the crystal lattice points which have identical geometrical surroundings. With u_j, v_j and w_j being fractions < 1, the primitive unit can be described by a set of basis coordinate vectors $\vec{r}_j = u_j\vec{a}_1 + v_j\vec{a}_2 + w_j\vec{a}_3$, locating all the j constituents of the unit cell with respect to the associated Bravais lattice point. All atoms in the crystal can then be reached by the vector $\vec{R}_{hkl} + \vec{r}_j$.

The angles between the primitive vectors are denoted (α, β, γ). Cubic structures have $\alpha = \beta = \gamma = 90°$ and equal length of the primitive vectors $|\vec{a}_1| = |\vec{a}_2| = |\vec{a}_3|$, simply denoted a. Within the cubic crystal system, there are Bravais lattices called *simple cubic* (SC), *face-centered cubic* (FCC) and *body-centered* (BCC), referring to the placement of lattice points, as illustrated in Figure 2.1. With different combinations of angles (α, β, γ) and primitive vector lengths $(|\vec{a}_1|, |\vec{a}_2|, |\vec{a}_3|)$, in 7 different crystal systems can be described containing in total 14 different Bravais lattices, see e.g. [24] pp. 26–28 for a complete overview.

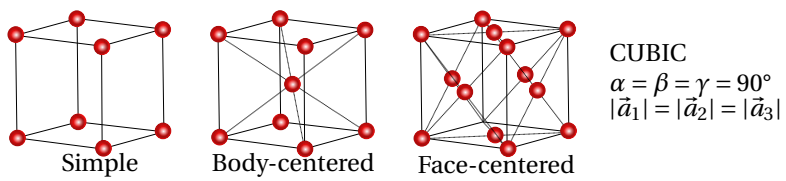


Figure 2.1: Schematic illustration of the cubic crystal system, 3 of the 14 Bravais lattices, see [24] pp. 26–28 for a complete overview. Figure adapted from here.

A lattice plane is denoted by (hkl), where the integers h, k and l are called *Miller indices*. By convention, the indices are written in lowest terms, that is, with 1 as greatest common divisor. The indices refer to the reciprocal of the intersections between the plane and the three axes spanned out by the primitive vectors $\vec{a}_i$, see Figure 2.2. A negative index is denoted with a bar above the number. A set of equal planes are denoted with curly brackets, $\{hkl\}$. For instance, the six sides of a SC structure are all equivalent by symmetry, and would be denoted $\{100\}$.

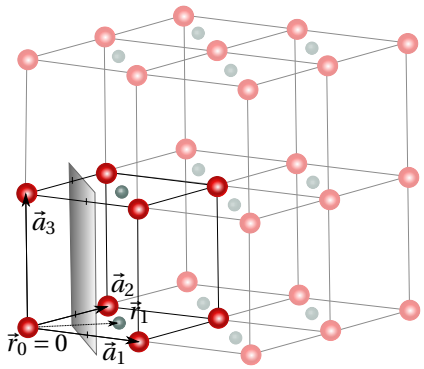


Figure 2.2: A simple cubic (SC) unit cell with basis $\vec{r}_0 = 0$ (red atom) and $\vec{r}_1 = 0.5\vec{a}_1 + 0.5\vec{a}_2$ (grey atom) in a Bravais lattice. Each point in the lattice is filled with a red atom. The indicated plane cuts the axes at $a_1 = 0.5$, $a_2 = 0.5$ and $a_3 = \infty$ with resiprocals $\frac{1}{a_1} = 2$, $\frac{1}{a_2} = 2$ and $\frac{1}{a_3} = 0$. Dividing by 2 gives the Miller indices $h = 1$, $k = 1$ and $l = 0$.

Directions within the lattice are denoted by vectors $[hkl]$ which are parallel to $\vec{R}_{hkl}$ of Equation (2.1). When viewing a crystal lattice along some specific directions, the atoms will line up in *columns*, referring to a row of atoms along the viewing directions. These directions are termed *zone axes* of *high-symmetry directions*. As an example, Figure 2.3 shows an FCC lattice observed in the [100], [110], [111] and [456] directions, with lower indices resulting in more space between atom columns normal to the view direction.

The interplanar spacing d in a cubic structure with lattice parameter a is given by

$$d = \frac{a}{\sqrt{h^2 + k^2 + l^2}}. \tag{2.2}$$

The angle ϕ between two planes can be found from the dot product of the normalized

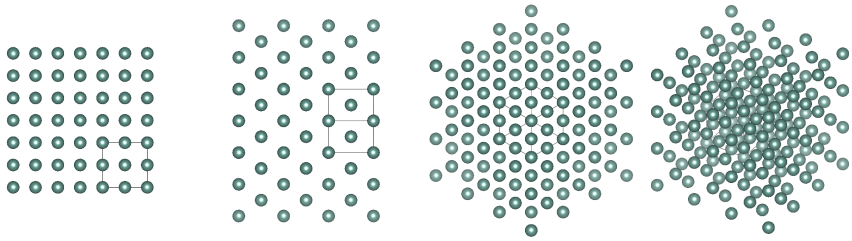


Figure 2.3: An FCC lattice observed in the high-symmetry directions [100], [110], [111] and [456], from left to right, showing how low-index orientations have a bigger distance between atom columns normal to the view direction.

normal vectors, $\hat{n}_1$ and $\hat{n}_2$, as given by

$$\cos\phi = \hat{n}_1 \cdot \hat{n}_2 = \frac{h_1 h_2 + k_1 k_2 + l_1 l_2}{\sqrt{h_1^2 + k_1^2 + l_1^2}\sqrt{h_2^2 + k_2^2 + l_2^2}}. \tag{2.3}$$

2.2 Grain Boundaries in a Polycrystalline Material

A polycrystalline material is an aggregate of several single crystals, or grains, of different size and orientation. Sizes can vary from a few nanometers to several millimeters[6]. The region between these crystals is called the *grain boundary* (GB). This region is interesting because it has important ramifications for the macroscopic properties of the material, whether it is metallic, intermetallic or ceramic[26]. For example, GB resistivity is observed to be at least one order of magnitude larger than in the bulk in some ceramics[21]. Furthermore, mechanical characteristic like deformation resistance[27] and crack propagation[28] is thought to be directly linked to the specific structure and type of bonding in the GB region. Studies have shown a close relation between the GB structure and ionic conductivity[29] and with thermal stability[30]. In the design and development of new high performance materials, one seeks to tailor the type of grain boundaries occurring in order to control the macroscopic properties. To achieve this, fundamental knowledge about the GB effects is essential, which is the reason why this has been extensively studied by researchers during the past century[6].

2.2.1 Grain Boundary Characterization

Grain boundaries can be divided into categories depending on the misorientation angle between the two adjacent grains. The main types are the *tilt* GB and the *twist* GB, for which the rotation axis is parallel and perpendicular to the GB plane, respectively, as illustrated in Figure 2.4. In nature, a mix of the two types is the most common. Considering whether the misorientation angle is below or above about 15°, the grain boundary is said to be a *low-angle* or *high-angle* grain boundary (LAGB or HAGB). For HAGBs, some specific angles lead to atomically well-matched grain boundaries with low interfacial energy, making them particularly stable[31]. These grain boundaries occur when the grains are rotated so that some lattice sites overlap, forming a bigger *coincidence site lattice* (CSL)[32]. They are therefore referred to as CSL grain boundaries. The GB atoms form a periodic pattern of so-called GB *units*. Figure 2.5(**(a)-(d)**) shows some examples of symmetric tilt HAGBs with GB units indicated by overlaid atoms. In 2.5**(e)**, a LAGB with $2\theta = 5°$ is shown. A periodic array of dislocations are formed to compensate for the strain induced by the misorientation, but no characteristic GB units are formed as for the HAGB.

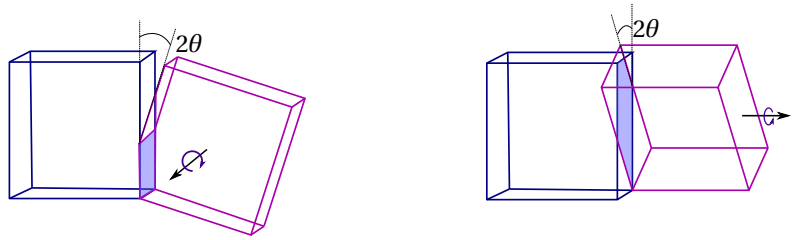


Figure 2.4: Illustration of how crystal grains are combined in a **(a)** tilt and **(b)** twist grain boundary with misorientation angle 2θ. The rotation axis is parallel and normal to the GB plane (blue), respectively.

The Σ value of a CSL GB is defined as the ratio of the total number of lattice sites to the number of coincidence sites, as illustrated in Figure 2.6. A low Σ value is therefore corresponding to a GB with a high number of common lattice sites, or common atoms. An LAGB is Σ1, as all atom sites in the GB are shared. It should be noted that the CSL theory is a simplified geometrical description. In reality, if it is energetically favorable,

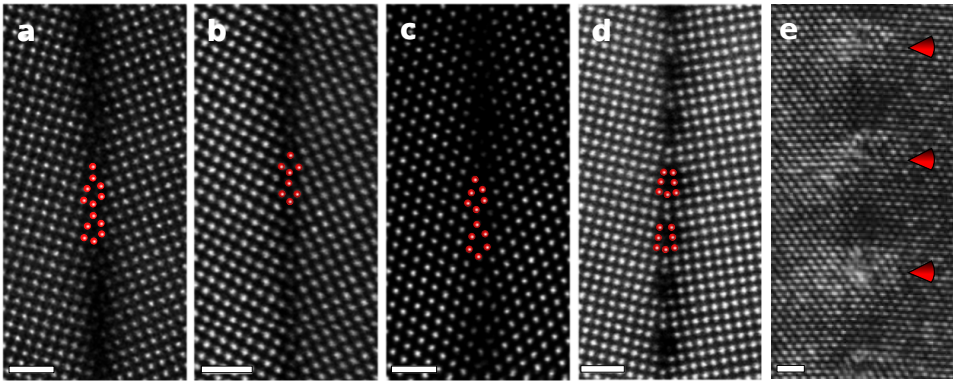


Figure 2.5: Examples of symmetric tilt HAGBs and LAGB in yttrium-stabilized zirconia (YSZ). **(a)**- **(d)** are HAADF-STEM images of CSL HAGBs with 2 GB units indicated by overlayed atoms, Σ5{310}/[001] GB, Σ5{210}/[001]/ GB, Σ9{221}/[110] GB and Σ{510}/[001] GB, respectively. **(e)** is a HRTEM image of a $2\theta = 5°$ LAGB with dislocations indicated by arrows appearing periodically along the GB plane. Scale bars are 1 nm. Images are reprinted with permission from [33] (**(a)**- **(d)**) and [34] (**(e)**).

relaxation mechanisms will lead to rearrangement of the GB atoms[31].

For a symmetric pure tilt GB, the orientation of the crystals are equal, only mirrored about the GB plane. In this case, when referring to a specific GB geometry, it is common to specify the Σ value, the tilt axis (rotation axis in Figure 2.4) and the GB plane. As an example, a symmetric tilt Σ5{310}/[001] in a cubic crystal structure has a GB plane with Miller indices 3,1 and 0, and the tilt axis [001] is parallel to the third primitive vector $\vec{a}_3$. The misorientation angle can be found from Equation (2.3) with $\hat{n}_1 = [310]$ and $\hat{n}_2 = [3\bar{1}0]$ as normal vectors to the GB plane in each grain, giving $2\theta = 36.87°$.

2.2.2 Yttrium Stabilized Zirconia

For the electrolyte in electrochemical ceramic devices such as solid oxide fuel and electrolyser cells (SOCs), high ionic conductivity and electrical resistivity are important features determining the overall performance of the device. One widely used material that fits these characteristics is Yttrium-stabilized cubic Zirconia (YSZ)[18].

At room temperature, pure zirconia is stable in a monoclinic phase. At elevated tem-

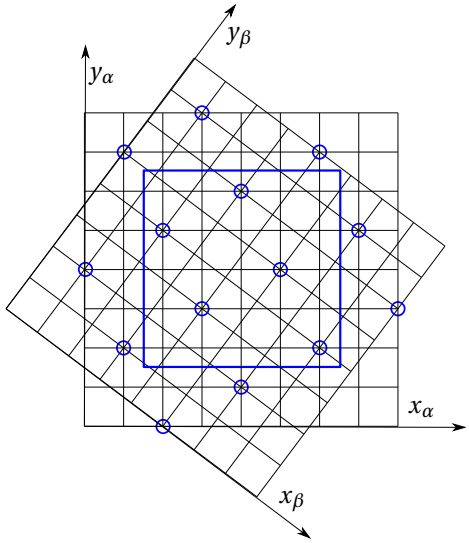


Figure 2.6: Illustration of the Σ notation. In this pure twist GB, lattice β is rotated an angle 2θ = 36.87° with respect to lattice α about the tilt axis normal to the paper plane. The blue square contains 25 lattice sites from each lattice, 5 of which are overlapping (marked with blue circles). This gives a ratio of 5 and therefore a Σ5 CSL GB. Figure adapted from [31].

peratures, transitions into tetragonal (1170°) and cubic (2350°) phases are observed, accompanied by significant volume changes[35]. To avoid this behaviour, zirconia can be stabilized in its cubic phase by adding covalent or trivalent dopant atoms before cooling it down to room temperature. For zirconia to be used in SOFs, yttria (Y_2O_3) is the most widespread choice[19]. Yttrium ions (Y^{3+}) substitutionally replace zirconium ions (Zr^{4+}) in the cation lattice, as illustrated in Figure 2.7. In this process, oxygen vacancies V_O of O^{2-} are generated to maintain charge neutrality, one per two Y^{3+}. These oxygen vacancies are the key factor facilitating the oxygen ion conductivity in YSZ[36]. In Kröger-Vink notation[37], the reaction is written

$$Y_2O_3 \longrightarrow 2\,Y'_{Zr} + V_O^{\bullet\bullet} + 3\,O_O^{\times}. \tag{2.4}$$

Here, Y'_{Zr} denotes an Y atom in a Zr lattice site with a single negative charge, $V_O^{\bullet\bullet}$ is an oxygen vacancy with double positive charge while $O_O^{\times}$ are O atoms in O lattice sites with a neutral charge. Consequently, YSZ has a cubic fluorite structure, with the cations (Zr^{4+} and Y^{3+}) forming a face-centered cubic (FCC) lattice, while the anions (O or V_O) occupy all tetrahedral sites. In the notation established above, this corresponds to an

FCC Bravais lattice with basis of one cation at $\vec{r}_1 = 0$ and one anion at $\vec{r}_2 = a[\frac{1}{4}\frac{1}{4}\frac{1}{4}]$ and $\vec{r}_3 = -a[\frac{1}{4}\frac{1}{4}\frac{1}{4}]$.

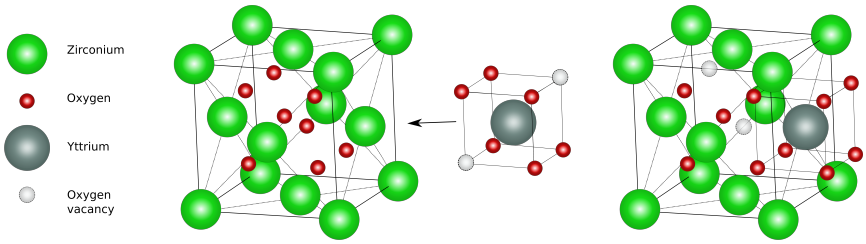


Figure 2.7: Adding Yttria (Y_2O_3) to Zirconia (ZrO_2) stabilizes the cubic structure. Dopant atoms substitutionally replace Zr in the cation lattice. Oxygen vacancies are introduced to maintain charge neutrality, facilitating oxygen ion conductivity.

Grain Boundary Segregation

A grain boundary is an extended structural defect, creating a difference in the standard chemical potential between the dislocation core and the bulk (grain interior). This gives rise to the phenomena called *grain boundary segregation* (GBS), referring to how dopant atoms tend to move towards the GB region in some GB types. An example of direct observation by EDX of yttrium segregation to the GB in YSZ is shown in Figure 2.8, as reported by Feng et. al. in [38]. The oxygen vacancy distribution is found to also be affected, and as this is the key parameter permitting ionic transport within the material, GBS has great implications for YSZ as an electrolyte material[33, 39]. As the macroscopic geometry of the GB is found to directly affect the segregation process, detailed knowledge about the 3-D structure in the GB region is essential to fully understand the segregation behavior and its influence on properties in zirconia ceramics[34].

2.2.3 Theoretical Calculations

To understand the nature of a grain boundary, one must study the collective behavior of the atoms in the near-GB region. The atom-atom interactions determine both atomic structure and physical properties of a material, e.g. how atoms in a crystalline material prefer to arrange themselves in particular highly regular grids, or the creation of periodic dislocations to compensate for internal strain. A general approach is to create a simple analytic function describing the potential energy of a set of atoms N atoms,

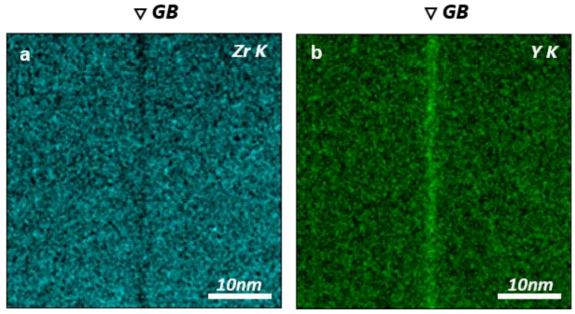


Figure 2.8: EDX maps from a Σ3{111}/[110] GB in YZS, showing **(a)** the Zr K map and **(b)** the Y K map. The GB is indicated by the arrow. The intensities are homogeneous in the bulk region, showing that the Y^{3+} are homogeneously distributed. For the GB region, the Y intensity increases while Zr intensity decreases, indicating GB segregation. Figure reprinted from [38] with permission (Supplementary Figure 1).

$V(r_1, r_2, ..., r_N)$. The force $\vec{f}_i$ on an atom i is then given by the negative derivative of this potential with respect to its position[40],

$$\vec{f}_i = -\frac{\partial V(r_1 ... r_N)}{\partial \vec{r}_i}. \tag{2.5}$$

The parameters of this potential function can be estimated either from analysis of suitable experimental data or from *first principles simulations*.

Calculations from first principles or *ab inito* refer to starting from the very fundamental interatomic interactions. A rigorous treatment of an atom-atom interaction involves considerations of the quantum mechanical motion and interaction of electrons, and should consequently be based on a solution of Shrödinger's equation describing the time-evolution of a physical system. Among the first principles methods is density functional theory (DFT)[16] in which the electron density is modelled by functionals. Such calculations are very computationally demanding, and are in practice feasible only for systems consisting of a relatively small amount of atoms. Still, is it important to appreciate how this theory provides the basis for constructing simplified, but efficient models needed for investigating large-scale problems.

Given a model of the interatomic potentials, bulk properties of a material or surface and dislocation phenomena can be simulated. In atomistic simulations based on *Monte*

Carlo (MC) methods, one seeks to mimic the random nature of a collection of interacting atoms in thermal equilibrium, as described by Bolztmann's statistics[41]. This describes the probability of finding a system in a specific state or configuration depending on its energy. Random moves are suggested, and the acceptance probability P depends in the resulting change in total energy ΔE_{tot} of the atom configuration[22],

$$P = \begin{cases} 1 & \Delta E_{tot} < 0 \\ exp\left(-\frac{\Delta E_{tot}}{k_B T}\right) & \Delta E_{tot} \geq 0, \end{cases} \tag{2.6}$$

where k_B is Boltzmann's constant and T is the system temperature. While the Monte Carlo methods create artificial trajectories, *molecular dynamics* (MD) tries to simulate "true" dynamics, in principle by numerical integration of Newton's third law relating the force $\vec{f}_i$ on an atom of mass m to the resulting acceleration $\vec{a}_i$ in the familiar equation $\vec{f}_i = m\vec{a}_i$, all while maintaining energy conservation[17].

Atomistic simulation has proven useful in studying the grain boundary segregation phenomena by investigating the driving forces and identifying energetically favorable spatial configurations. Through simulations with MC[42] or an MC-MD combination[43] for a symmetric tilt $\Sigma 5\{310\}[001]$ GB model in YSZ, dopant Y atoms are found so segregate preferentially to specific GB sites. Such findings can be further supported if experimental data from the same structure is available. For the results to be directly comparable to experiments, however, it is important that the exact same structure is considered.

The choice of GB to study is often one of the geometrically simple CSL GBs introduced earlier, as these contain a repeating pattern of dislocations so that calculations over only one of the units is necessary, hence confining the problem to a relatively small region. A small structure cell is essential in theoretical calculations to reduce the computational cost. In addition, periodic boundary conditions can be constructed, removing the effect of artificial surfaces otherwise affecting the calculation results[17].

In all of the techniques discussed so far, an initial model resembling a realistic GB configuration is essential. The routine approach is to manually rotate, cut and paste to-

gether to grains, each with an ideal lattice structure resembling the bulk of the material one wishes to study (see e.g. [44, 43, 38, 34, 42]). Subsequently, rigid body translations are performed, i.e. adjusting the relative position of the grains, and atoms that end up too close are deleted. After this, the most likely position of each individual atom is calculated by minimizing the total structure energy. A variety of options for energy minimization algorithms exits, including static lattice calculations, genetic algorithms or simulated annealing (see e.g. [17]). The success of this step, and hence all the following simulation results, depends on a good initial structure. Ideally, an initial structure should be obtained directly from experimental images, as this would make calculation results directly comparable to experimental observations.

2.3 Transmission Electron Microscopy

A transmission electron microscope (TEM) makes use of the dual particle-wave behavior of electrons to generate images, diffraction patterns and different kinds of spectra from a specimen, providing high-resolution structural and analytical information. With its versatility and atomic resolution available for some operational modes, TEM has become an indispensible tool in material science, condensed matter physics and engineering[8]. In this section, the physical concepts behind the microscope are discussed, with focus on high-angle annular dark-field scanning TEM (HAADF-STEM).

2.3.1 The Transmission Electron Microscope

Much of this section is based on the book "Transmission Electron Microscopy" by Williams and Carter[45]. Citations are omitted in this section for readability.

In some ways, a TEM can be compared to the familiar optical microscope, as many of the principles they operate through are similar. The ultimate goal is the same; to achieve a clear image of the object being investigated. There are however some important differences, making the TEM much more complex and versatile. In the optical microscope, visible light is directed through curved glass lenses, which focus the light by means of refraction. In TEM, electrons are used in stead of light, and the focusing

is done by electromagnetic coils creating a magnetic field interacting with the charged electrons by means of the magnetic Lorenz force. Another main characteristic of the TEM lies in the word *transmission*. The electrons forming the image is transmitted through, instead of reflected by, the specimen. The electrons simultaneously display both wave and particle characteristics when interacting with the sample, generating a variety of signals that can reveal information about the structure and chemical composition of the sample at hand. The concept of matter behaving like waves was first proposed by Louis de Broglie in 1924. The wavelength λ and the momentum p of a massive particle are related by the Planck constant h as follows,

$$\lambda = \frac{h}{p}. \tag{2.7}$$

Inside the TEM, the electron is accelerated by a voltage V, giving it a kinetic energy $E_k = eV$ where e is the elementary charge. At operation voltages above 100 kV, the speed of the electrons is more than half the speed of light. Therefore, relativistic effects must be taken into account. The total energy E of the particle can be expressed in terms of its momentum p, its rest mass m_0 and the speed of light c,

$$E = \sqrt{p^2c^2 + m_o^2c^4}. \tag{2.8}$$

The total energy can also be expressed as the kinetic energy plus the rest energy,

$$E_k = E - m_0c^2. \tag{2.9}$$

Combining this with Equation (2.7), we get the following expression relating the applied voltage V with the wavelength of the electron,

$$\lambda = \frac{h}{\left[2m_0E_k\left(1 + \frac{E_k}{2m_0c^2}\right)\right]^{1/2}}. \tag{2.10}$$

Similarly to the image resolution of an optical microscope, which is limited by the wavelength of visible light as stated by the classic Rayleigh criteria, the wavelength associated with the high-energy electrons is the limiting factor for the resolution in TEM. There-

fore, a similar criteria exists for the theoretically best resolution δ achievable using electrons, stated in terms of the associated wavelength as follows[46],

$$\delta \approx \frac{1.22\lambda}{\beta}, \tag{2.11}$$

where β is the semi-angle of collection of the magnifying lens. However, an imperfect electron source leads to a spread in the energy of the emitted electrons, leading to a resolution lower than what is promised by Equation (2.11). Furthermore, intrinsic imperfections of lenses gives rise to chromatic and spherical aberrations, with names referring to the analogy with aberrations in optical lenses. The effects of imperfections in the electron source can be reduced by energy filtering, and apertures reducing the width of the beam can be used to partly compensate for the poor lenses. Moreover, with the recent development of special lenses correcting the aberration, sub-Ångström resolution is now routinely available. The current record for spacial resolution is 40.5 pm[9], achieved by Morishita et. al. with a 300 kV aberration corrected STEM.

2.3.2 Image Formation in STEM

Developed by Crewe and coworkers[47], scanning TEM (STEM) is a special TEM operation mode where a focused electron beam, or *probe*, is scanned across the sample in a line-by-line, pixel-by-pixel manner, dwelling a millisecond or two at each site. This gives a serial data acquisition, as opposed to in conventional TEM, where a stationary parallel beam illuminates the sample. In state-of-the-art instruments, the electron probe can have atomic-scale dimensions, wich is part of what enables the high spatial resolution[48]. The STEM operation mode gives rise to a wide range of signals. Figure 2.9 summarizes the most important signals providing information on the sample chemistry and structure. Spectroscopic images can be formed by measuring energy loss in the transmitted electrons (electron energy loss spectroscopy (EELS)[49]) or by registering the characteristic X-rays emitted from the sample (energy-dispersive X-ray spectroscopy (EDX), demonstrated at atomic resolution in [50]). These techniques enables the electron configuration of the constituent elements and chemical composition of the sample or to be revealed. Either way, the scanning technique with a focused probe in STEM provides a straight-forward way to relate the measured signal to specific sites in

the specimen.

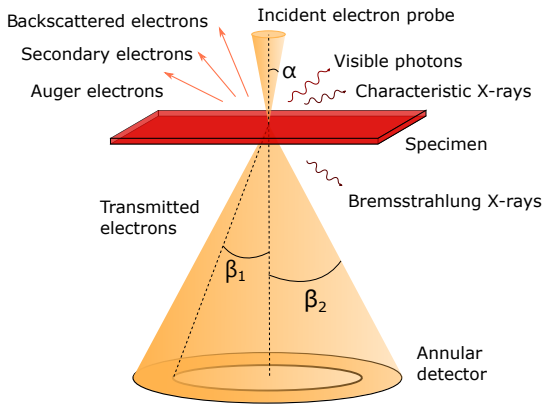


Figure 2.9: Some of the signals generated in STEM. α is the probe-forming aperture semi-angle, while β_1 and β_2 are inner and outer detector angles, respectively. Figures combined and adapted from [45] p. 7 and p. 380.

Electron *scattering* refers to how the negatively charges electron may be deflected by electrons or the nucleus of atoms in the specimen[45]. The scattered electron beam might be coherent or incoherent, depending on the nature of the electron-specimen interaction. Annular detectors of with specific inner and outer angles (β_1 and β_2 in Figure 2.9, respectively) can be used to selectively detect parts of the signal. Typical ranges of detector angles are 0–10 mrad for bright field (BF), 10–50 mrad for annular dark field (ADF) and above 50 mrad for high-angle (HA)ADF detectors[45]. For BF imaging, mainly coherently scattered electrons are detected, resulting in so-called phase-contrast images, where the contrast is dependent on the phase associated with the wave function of the electrons in the beam[48]. The wave functions might interfere, leading to an unintuitive image contrast that must be interpreted by means of simulation[51]. The idea of leaving the coherently scattered electrons and only collect incoherently scattered electrons at higher angles was proposed by Howie[52], leading to the development of the high-angle annular detector. With a HAADF detector, the image is formed by mainly collecting the electrons scattered incoherently (Rutherford scattering and thermal diffuse scattering (TDS)). The result, a so-called *Z-contrast image*, is highly sensitive to the atomic number of the atoms in the sample[53]. Unlike in

phase-contrast imaging, this technique provides an intuitively interpretable image, as no reversals in image contrast occur[54].

The probability that an electron is scattered through an angle θ into a solid angle dΩ depends on the *scattering cross-section* of the atoms it encounters. The Rutherford scattering cross-section is defined as follows[45],

$$\sigma_R(\theta) = \frac{Z^2 \lambda_R^4}{64\pi^4 a_0^4} \frac{\mathrm{d}\Omega}{\left(\sin^2\left(\frac{\theta}{2}\right) + \frac{\theta_0^2}{4}\right)} \tag{2.12}$$

Here, Z is the atomic number of the scattering species. Relativistic effects are taken into account by using the relativistic wavelength of the electron λ_R and the Bohr radius a_0 ($\approx 0.0529\,\mathrm{nm}$). With E_0 as the electron energy in keV, $\theta_0 = \frac{0.117 Z^{1/3}}{\sqrt{E_0}}$ describes how the negative electron cloud screens the positively charged nucleus. If the scattering angle is higher than θ_0, the electron-electron interaction can be neglected. In that case, the Rutherford cross-section, and hence its contribution to the intensity of the signal I, is proportional to Z^2, as seen in the equation above. At lower angles, where the screening of the nucleus can not be neglected, the intensity dependency ranges from Z^2 down to $Z^{3/2}$[55]. While the Rutherford scattering does not sufficiently describe the wave-nature of the electron beam, which is relevant for the signal intensity at low angles, it is useful when considering high-angle scattering. Equation (2.12) is however only valid at operating voltages lower than 300–400 keV and for lighter elements ($Z < 30$)[45].

In addition to Rutherford scattering, thermal diffuse scattering (TDS) (electrons scattered by phonons) also contributes to the intensity in a Z-contrast image[8]. This scattering depends on the Debye-Waller factor, varying with atom species as well as structure configuration[56, 57]. The relation between signal intensity and atomic number is therefore better described by $I \propto Z^\gamma$ where γ is a factor that depends on the observed sample, but also on the specific experimental conditions. As an example, with 24.5 mrad probe semi-angle, 24.5 pA probe current and HAADF detector range of 54–270 mrad, the atomic column intensity is approximately proportional to $Z^{1.7}$[58].

The incoherent image formation in HAADF-STEM can be modelled as follows. With the STEM probe at position $\vec{r}_{i,j} = [x_i, y_j]$, the expected image intensity $f_{i,j}$ in the pixel (i, j) can be written mathematically as a convolution between an object function, representing the crystal lattice of the specimen, and the probe intensity function as follows[53],

$$f_{i,j}(\rho) = f(\vec{r}_{i,j};\rho) = O(\vec{r}_{i,j};\rho) * P(\vec{r}_{i,j}) \tag{2.13}$$

Here, $O(\vec{r}_{i,j};\rho)$ is the object function, which depends on the structure parameters ζ, and $P(\vec{r}_{i,j})$ the probe intensity function. The crystal lattice can be modelled as a superposition of Gaussian peaks, one for each atom column, giving the following object function,

$$O(\vec{r}_{i,j};\rho) = \kappa + \sum_{m=1}^{M} \frac{A_m}{2\pi\sigma_{x_m}\sigma_{y_m}} e^{-\left(\frac{(x-\mu_{x_m})^2}{2\sigma_{x_m}^2} + \frac{(y-\mu_{y_m})^2}{2\sigma_{y_m}^2}\right)}. \tag{2.14}$$

Hence, the unknown structure parameters denoted by ζ are the amplitudes A_m,standard deviations $(\sigma_{x_m}, \sigma_{y_m})$ and the peak centers (μ_{x_m}, μ_{y_m}) corresponding to atomic column positions. κ is a constant background. The probe intensity $P(\vec{r}_{i,j})$ depends on probe parameters like the acceleration voltage, aberration coefficients, defocus and the objective aperture semi-angle.

With this incoherent imaging model, there is no interference between electron waves scattered from different atoms or atomic columns. The intensity in each column can therefore be interpreted separately, allowing for quantitative image analysis methods. This assumes a strong *channeling effect*, referring to how the electrons in the beam are thought to propagate through the sample. When the probe is located above an atom column it "channels" down the column, leading to a peak in the signal intensity at this position. When located between columns, the beam spreads, leading to a reduced intensity[59]. This has proven to be a safe assumption in zone-axes directions where the atom column distance is sufficiently big. However, if so-called *cross-talk* is present, the intensity in one column is no longer independent of the intensity of others. While the channeling usually ensures that the probe is focused along one atom column at a time, some intensity might be transferred to neighboring columns in thicher samples, or for a higher-index zone axis where atomic columns are too close[48]. As an example, in [60],

Van Aert et. al. use multi-slice image simulations[55] to investigate the phenomenon in aluminum. They show that the assumption of independent column intensities is valid for two low-index zone axis, [100] and [10$\bar{1}$], up to a sample thickness of about 50 nm, while cross-talk was notable even at 10 nm for zone axis [411].

Random Noise

The deviation from the theoretically expected image intensity described by Equation (2.13) can be described approximately by the addition of a stochastic function to the object function. By doing so, the noise is assumed to be additive. This can be done only if the noise is independent of the expected signal. This is a valid assumption for the so-called shot noise, originating from fluctuations in the electron beam due to the random nature of electrons.[61]. The number of electrons N hitting a specific area during the time T follows the Poisson distribution with a standard deviation of nearly $\sqrt{N}$ A better signal-to-noise ratio (SNR) can therefore be achieved by averaging over many images of the same object.

Image Distortions

The sequential pixel-by-pixel image acquisition technique of STEM provides an intuitive image interpretation. However, in combination with atomic scale magnification, the price to pay is a high sensitivity to movements of the sample, as induced by disturbances in the instrument or the environment like mechanical or acoustic vibrations or electromagnetic interference[62]. If these factors result in sample drift, the recorded image may appear stretched or skewed, and therefore exhibiting unreliable lattice parameters. In the case of a constant or sufficiently slowly varying drift rate[1], simple geometric considerations might be used to compensate for the drift, e.g. by the use of an Affine transformation estimated from the similarity between the distorted image and reference image generated from a priori knowledge of the sample[64]. Details of this technique is found in Appendix A. This is similar to what is done in [65], where a polynomial warping of a distorted HAADF-STEM image is estimated from a quantitative high-resolution TEM (HRTEM) image of the same sample area. If the drift rate is not

[1] Drift rate change $\approx < 0.005$ Hz is sufficiently slow, according to [63]

continuous, non-rigid registration must be considered, see e.g. [63, 58]. A slow scan (longer total frame time) makes the sample drift distortions more prominent, hence, a fast scan is desirable.

2.4 Tomography

Over the last 50 years, 3-D reconstruction from projections is a problem that has gained interest in a large variety of scientific fields. One important version of the problem is found in medicine, where the emergence of computed tomography (CT) has revolutionized the field of diagnostic radiology since its development began in the early 1970's[66]. In a CT scan, multiple X-ray projection images are made, from which the density distribution within the human body is obtained. The same fundamental reconstruction process has been utilized in many fields within medicine, as well as in science and engineering. The resolution of X-ray tomography is however limited by the wave-length of the X-rays. In combination with poor lenses, the resolution has never exceeded 2 µm[67]. However, tomographic reconstruction at atomic resolution is under development using HAADF-STEM, providing new applications in 3-D material characterization. In this section, the mathematical background of tomography and tomographic reconstruction is introduced, along with examples of its application in material structure characterization.

2.4.1 Tomographic Imaging and Reconstruction

The main challenge of tomographic reconstruction is to obtain an estimate of the density distribution within an object based on a finite number of projections. Johann Radon laid the mathematical foundation for tomographic imaging already in 1917[68], introducing what is later known as the *Radon transform* R. If a function $f(x, y)$ represents the intensity distribution in an image, the Radon transform of the image would correspond to projections, or line integrals across the image from all directions, along all possible lines L with unit length ds,

$$Rf = \int_L f(x, y)\,\mathrm{d}s. \tag{2.15}$$

The inverse Radon transformation would bring back the original image. However, it is

evident that in practice, the experimental projections of the object will be discrete, and so forth the reconstruction ends up imperfect. Aim of tomographic reconstruction is therefore to provide the best possible estimation of the original image from the limited information available[66].

The geometry of the problem is illustrated in Figure 2.10, giving the following relation for a line traversing the imaged object through the coordinates $(x,\ y)$ at an angle θ and distance s from the origin,

$$s = x\cos\theta + y\sin\theta. \tag{2.16}$$

From this, we can find the following expression for the projection line,

$$y(x;s,\theta) = -x\cot\theta + \frac{s}{\sin\theta}. \tag{2.17}$$

The projection $p(s,\theta)$ gives the value of the line integral of the object density distribution function $f(x,y)$ along this line.

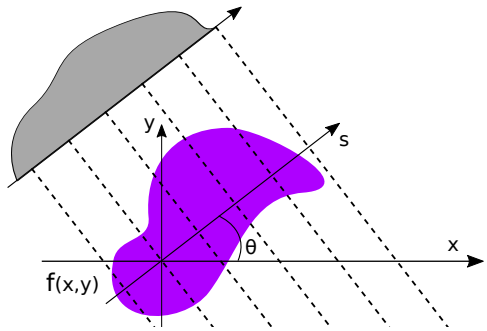


Figure 2.10: Visualization of the Radon transform. The function $f(x,y)$ represents the density distribution within the object. The gray area illustrates the values of the line integrals of the object in the direction normal to s. The line integrals for all possible lines at all angles θ gives the Radon transform Rf. Figure inspired by [67] p. 416.

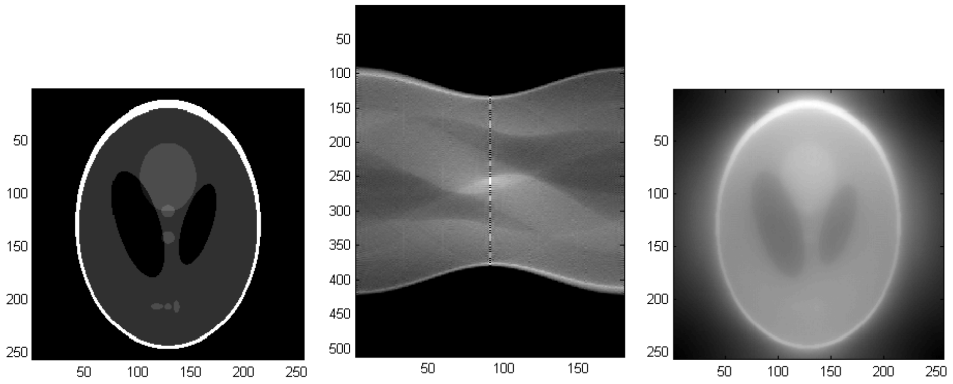
Direct Back-Projection

All available tomographic reconstruction algorithms are in some sense approximations of the inverse Radon transform. As a first guess, we can do nothing better than assuming that the density distribution $f(x,y)$ is evenly distributed across the object. This is called

direct back-projection (DBP). Given that the beam enters the object at t_1 and exits at t_2, this is given as

$$g(x,y;s,\theta) = \frac{p(s,\theta)}{t_2 - t_1}. \tag{2.18}$$

Doing the back-projection for all values of x, y, s and θ, we obtain the direct back-projected image g, a smeared-out version of the original image. Figure 2.11 shows this approach being applied to a simple phantom image directly available in Matlab. 180 projections are made with 1° increments. The collection of projections is called a *sinogram*, shown in Figure 2.11b, where one vertical column of pixels corresponds to one projection. The resulting DPB image appears blurry, but features of the original image are visible. Further on, several methods exist to improve the reconstruction result, depending on the application. One option is to filter the projected data prior to reconstruction, yielding so-called *filtered back-projection* (FBP) (see e.g. [69]).



(a) Original image **(b)** Sinogram **(c)** DBP image

Figure 2.11: Illustration of direct back-projection (DBP). Units of all axes are pixels. The original image in **(a)** is a phantom image available in Matlab. **(b)** shows the sinogram, where each vertical column of pixels corresponds to one projection of the original image, one for each projection angle. Each pixel in one vertical column can be interpreted as the integral of the image along one projection "ray", illustrated by stippled lined in Figure 2.10 for one angle θ. The result of direct back-projection is shown in **(c)**, appearing as a blurry version of the original image.

Iterative Reconstruction Algorithms

Tomographic reconstruction can be described mathematically as the reconstruction of an object x from its 2-D projections b, acquired by a projection operator A. An *iterative*

tomographic reconstruction method is then based on solving the following least-square minimization problem by incremental improvements,

$$\hat{x} = \arg\min_x \left\| Ax - b \right\|_2^2, \tag{2.19}$$

where $\|u\|_2 = (\Sigma_i u_i^2)^{1/2}$ is the l_2-norm of u. Starting with an initial 3-D reconstruction x_1, a 2-D re-projection Ax_1 is made, and compared to the original 2-D projection images b. The difference is called the *projection error*, and solving the problem stated in Equation (2.19) corresponds to minimizing this error. Examples of common algorithms that implement this in different ways are the simultaneous iterative reconstruction technique (SIRT)[70], the algebraic reconstruction technique (ART)[71] and its variant the simultaneous ART (SART)[72]. When the available projection data is noisy or limited, either in terms of number of projections or projection angle range, these iterative reconstruction algoritms have proven more suitable than the simple FBP[10].

3-D Tomography

A tomographic *tilt series* is obtained by rotating the sample about one or more *rotation axes*[2] and obtaining a series of 2-D projection images normal to these axes. If a single axis geometry is used, that is, projections are made while rotating the object about one axis only, the mathematical reconstruction problem reduces to a set of two- dimensional ones. The reconstruction can then be done in slices which are normal to the rotation axis (see e.g. [73]).

The Projection Criterion

Images obtained with a conventional visible light camera will have intensities depending on the surface of the imaged object, making them unsuitable for tomographic reconstruction. In order for 3-D reconstruction to be feasible, the images must contain information on the object interior as well. This is known as the *projection criterion*. This is the case in X-ray tomography, where the image intensity is proportional to the sample density and thickness. In ADF-STEM imaging, the contribution of an atom to the image

[2] *Tilt axis* is the common term for this - however, since this word is used earlier in this report to denote the orientation of grains in a grain boundary, the word *rotation axis* will instead be used.

intensity is proportional to Z^γ. To a first approximation, the image intensity is therefore proportional to an integral of Z^γ through the sample along the beam direction[74]. In both cases, the resulting image is a 2-D projection of the object at hand, meaning that both techniques are well-suited for 3-D reconstruction.

2.4.2 Tomography for 3-D Material Characterization

Several different techniques have been reported to give a successful tomographic reconstruction of crystal structures at the atomic scale. In particular, the atomic arrangement in *nanoparticles* has been widely studied, as their small size (in the nm range) makes them suitable from STEM observation from multiple directions. This has lead to the development of several approaches aiming for atom-by-atom structure determination. The method most similar to conventional tomography is termed atomic election tomography (AET), requiring a high number of projections over an as wide as possible range. In [11], a gold nanoparticle with a diameter of 10 nm was reconstructed from 55 projections, overcoming the missing wedge-problem in the data sampling through equally sloped tomography (EST)[75]. Not all atoms could be resolved, but information about lattice parameters and grain orientation could be obtained. A similar approach was used 13 years later, when Yang et. al. reconstructed a 8.4 nm FePt nanoparticle using 68 ADF-STEM images over a tilt range from $-65.6°$ to $64.0°$[15]. The generalized Fourier iterative reconstruction (GENFIRE)[76] algorithm was used, iterating between real and Fourier space while enforcing physical constraints like a tight particle boundary. This is found to tackle the missing wedge-problem well. The big difference in the atomic number of Fe ($Z_{\mathrm{Fe}} = 26$) and Pt ($Z_{\mathrm{Pt}} = 78$) leas to a big difference in intensity in the ADF-STEM images, making it possible to distinguish the atom types. This approach could not be applied to e.g. YSZ, as the atomic numbers $Z_{\mathrm{Zr}} = 40$ and $Z_{\mathrm{Y}} = 39$ leads to too little difference in image contrast for Zr and Y atoms.

While AET requires a high number of projections, other techniques are developed that utilize only a very few high-symmetry zone axis projections. In [14], Goris et. al. uses a method based on compressed sensing (CS) for reconstruction of a gold nanorod from only four zone-axis projections. Compressed sensing electron tomography (CS-ET),

which yields a sparse reconstruction with typically few nonzero elements, is described more in detail in [77]. Discrete tomography (DT) for electron tomography was developed and applied to a simulated gold nanoarticle in [78], giving an atomic resolution reconstruction from only 3 projection images. DT deals with the challenge of reconstructing images containing only a few pixel values[79, 80]. Algorithms are often highly problem-specific, designed to take into account as much prior knowledge as possible. Therefore, far fewer projections are needed, compared to conventional tomography. DT was also utilized in [60] for a real silver nanoparticle (diameter 3 nm) within an aluminum matrix from only two projections. The number of atoms in each column is counted based on the intensity of the HAADF-STEM image, before a DT algorithm is applied to find the most likely arrangement of the atoms. In this approach, all atoms are assumed to fit rigidly on a pre-determined crystal lattice. While this may be a good assumption for a particle embedded in a supporting matrix, the method would fail to reconstruct possible structural deviations for a free-standing particle. Moreover, for the atom counting to work, a simple relation between image intensity and sample density is required. This is not always the case, e.g. near crystal defects[81, 23].

Image Alignment

Good alignment of the projection images in the tilt series is crucial for achieving an accurate 3-D reconstruction. In electron tomography, one conventional alignment approach is based on fiducial markers[82, 67]. High contrast particles such as colloidal gold beads are fixed within the field of view together with the sample of interest, creating a reference throughout the data set for translational alignment as well as for correcting image rotation or magnification change that might be present in the tilt series. This method can however not achieve atomic precision due to the size of the markers[11].

Another common method is based on cross-correlation between consecutive projection images[83]. When applied to already rotationally aligned images, this process is straight-forward. First, the change of scale normal to the tilt axis is adjusted for, based on the known difference in image acquisition angle (tilt). Then, the potential rela-

tive translation simply corresponds to the peak of the cross-correlation function between the two images, indicating the highest similarity. The process is repeated for each pair of consecutive images. This method does however require a dense tilt series with small difference in acquisition angle, so that consecutive projection images are similar enough for the 2-D cross-correlation to give a meaningful result.

In [11], another approach for image alignment is implemented, based on the fact that the center of mass in all projection images should coincide with that of the 3-D imaged object. This method is also employed in [14] for a gold nanorod. This would however not work unless the boundary of the object is contained within each projection image, as for small nanoparticles.

None of the methods discussed here would provide more than a coarse initial alignment, which should futher be adjusted. This can be done iteratively by aligning the images to re-projections of the reconstructed structure[10].

Chapter 3

Experimental Data

In Chapter 4, the 3-D reconstruction method will be explained and demonstrated by applying it to HAADF-STEM images of a symmetric tilt $\Sigma 5\{310\}/[001]$ grain boundary in yttria-stabilized cubic zirconia (YSZ). In this chapter, it is therefore explained how the sample was prepared and how the experimental images were obtained. This work was performed by Bin Feng and Ryo Ishikawa, assistant professors at the Crystal Interface Laboratory at the University of Tokyo. In the following, the details of the experimental method are presented to give the reader a complete overview. The experimental images presented in the final section are the starting for the 3-D reconstruction. A clear idea about the sample geometry and how the images were obtained is important, as this is key to how we can interpret the final reconstruction results.

3.1 The Bicrystal Model

In ceramic material research, the fabrication of bicrystals is a common way to design grain boundary models with a specific geometry. Two single crystals are carefully cut and joined, e.g. by diffusion bonding[84], creating a well-bonded artificial grain boundary. The process is illustrated in Figure 3.1. This enables systematic studies relating the grain boundary properties to its geometry, and results can be compared to previous research in a more direct manner. When high-angle GBs are studied, the CSL GBs are a common choice as these provide a clearly defined GB structure. For atomic resolution

STEM observation, pure tilt GBs are the most suitable. The bicrystal method has been applied in several studies, see e.g. [85, 86, 87, 34, 38, 88].

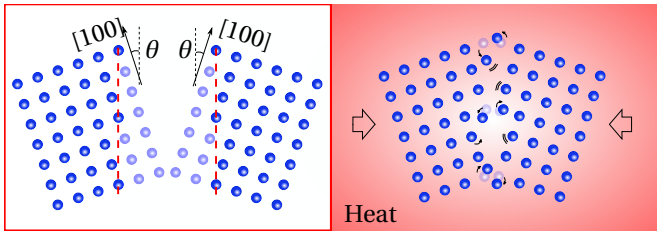


Figure 3.1: Illustration of the bicrystal method, creating a symmetric tilt Σ5{310}/[001] GB model. Two single crystals are rotated about the [001] tilt axis, $2\theta = 36.8°$, and cut at a {310} plane. During diffusion bonding at high temperature, the GB atoms will rearrange to lower the interface energy.

To obtain good experimental images, the crystal must be observed in high-symmetry directions so that atom columns are clearly distinguishable. The GB plane must be observed edge-on to avoid overlap of the two sides of the bicrystal in the projection images, as illustrated in Figure 3.2. The included table lists the crystal orientations for which images were obtained, along with corresponding rotation angle about the [310] axis normal to the GB plane.

3.2 Experimental Images and Details

3.2.0.1 Sample Preparation

The YSZ bicrystal studied in this thesis was fabricated by joining two single crystals of composition ZrO_2- 9.6 mol% Y_2O_3 at 1600 °C for 15 hours in air. From this, TEM sam-

Direction	Rotation angle β
[001]	0°
[1$\bar{3}$4]	38.3°
[1$\bar{3}$2]	57.7°
[1$\bar{3}$0]	90°

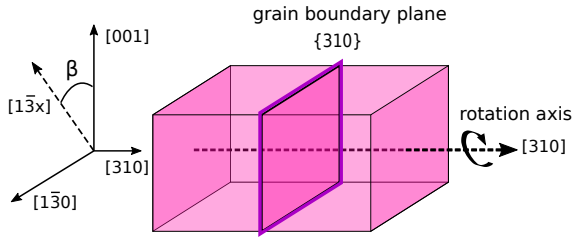


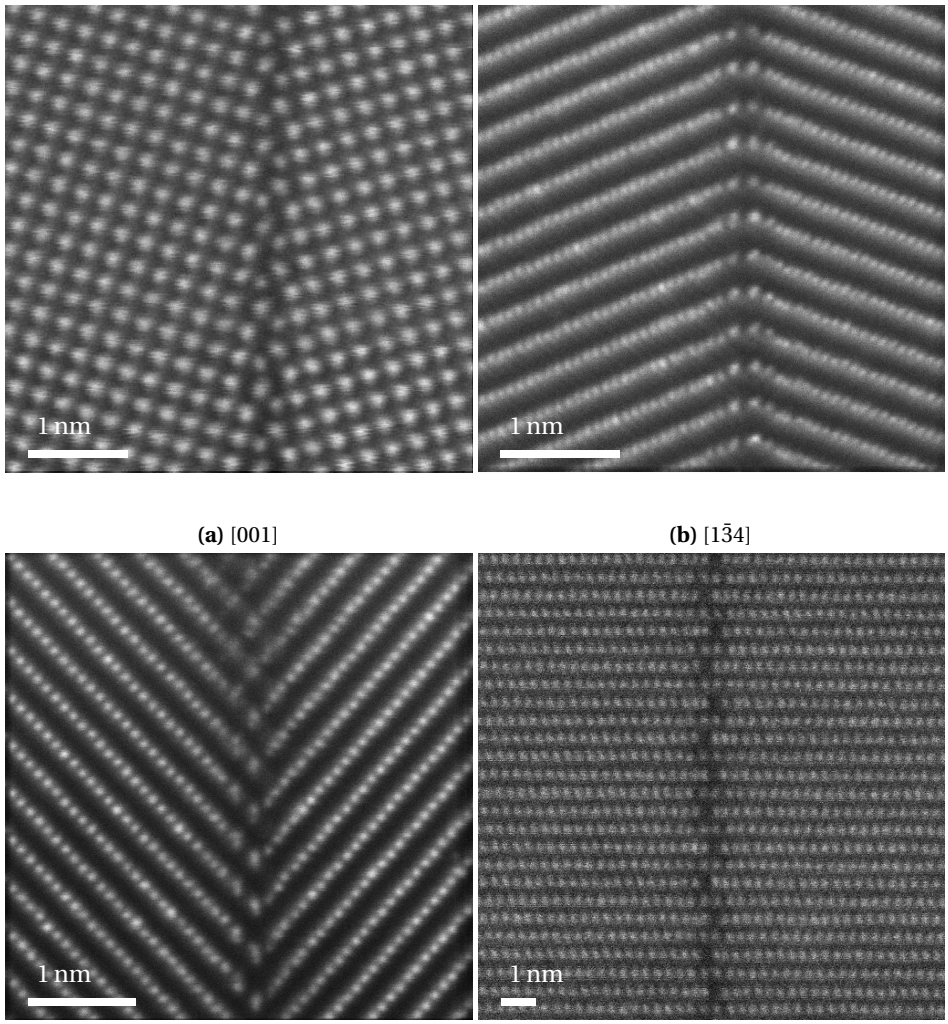
Figure 3.2 & Table 3.0: The bicrystal containing the GB model is rotated about the [310] axis and observed in four high-symmetry directions, always edge-on the {310} GB plane .

ples were prepared by first cutting thin slices of the bicrystal, so that the surface normal align with the desired observation direction. The samples were further thinned by mechanical polishing and Ar-ion beam milling, creating small holes along the GB. Near the edge of a hole, a suitable site for STEM observation can be found where the sample thickness is about 30 nm.

3.2.0.2 HAADF-STEM

High-angle annular dark-field (HAADF) STEM images were taken with the detection angle of 68–280 mrad (JEM-ARM200CF and JEM-ARM300F GRAND ARM, JEOL Co. Ltd). Several fast scans were performed to reduce effects from sample drift. Then the images were aligned using cross-correlation and averaged (same procedure as described in [89]). The resulting images are shown in Figure 3.3. Throughout the rest of the thesis, the images will be referred to as "Image $[hkl]$" with $[hkl]$ referring to the observation direction.

In HAADF-STEM observation, it is difficult to image light and heavy atoms at the same time, as the signal intensity scales approximately with Z^2. In YZS, with atomic numbers $Z_{\mathrm{Zr}} = 40$, $Z_{\mathrm{Y}} = 39$ and $Z_{\mathrm{O}} = 8$, the cations Zr and Y will clearly dominate. Observing the crystal along [001], the projected distance (normal to the view direction) between oxygen columns and neighbouring cation columns is 0.36 nm. Although this is within the resolution limit, the oxygen is not detectable due to the large tail form the zirconium scattering profile, as discussed in [23]. We therefore consider only the cation lattice for the 3-D reconstruction.



(a) [001] **(b)** [1$\bar{3}$4]

(c) [1$\bar{3}$2] **(d)** [1$\bar{3}$0]

Figure 3.3: HAADF-STEM images of the Σ5{310}/[001] GB in YSZ in several zone axis orientations, as indicated in the subcaption. Images were aquired with detection angle 68–280 mrad on JEM-ARM200CF and JEM-ARM300F GRAND ARM, JEOL Co. Ltd, by Bin Feng and Ryo Ishikawa.

Chapter 4

The 3-D Reconstruction Method

In this chapter, a method developed to reconstruct the 3-D atomic structure of a crystal grain boundary is presented. The method is based on tomographic reconstruction from atomic resolution HAADF-STEM images. In addition, all steps for preparing the experimental images are explained. Firstly, a brief overview is given. Thereafter, each step is explained more in detail, with examples from the application on the bicrystal model of a symmetric tilt $\Sigma 5[001]/\{310\}$ grain boundary in YSZ. Code is developed in Python for all image preparation steps, as well as the reconstruction process, and added in Appendix C. Details on the implementation are included in Appendix B, written as a tutorial with examples of how to use the code.

4.1 Work Flow

1. **Preparing the Experimental HAADF-STEM Images**

   (a) **Geometric correction of image distortions**
   Affine transformations are applied to the original experimental image to correct distortions from sample drift during imaging.

   (b) **Scaling**
   Finding the pixel/nm ratio from theoretical distance between atom planes.

(c) **Noise reduction by image averaging**

The images are averaged over several GB units to reduce noise. Optimal image template shifts are determined by cross-correlation.

(d) **Identify positions of atom columns**

Determining atom column positions by fitting a 2-D elliptical Gaussian function to each intensity peak.

2. **Iterative Tomographic Reconstruction**

(a) **Create initial 3-D structure from two images**

Tomographic reconstruction by direct back-projection

(b) **Image alignment refinement**

Image alignment by iterative re-projection and reconstruction

(c) **Add another image**

A third image is added and aligned to the re-projection of the reconstruction from two images

(d) **Reconstruction from three images**

Step 2a and Step 2b are repeated, with the image alignment refinement applied to the third image only

(e) **Reconstruction from more images**

Step 2c and onward are repeated to add more images, one by one.

4.2 Part 1: Preparing the HAADF-STEM Images

An accurate 3-D reconstruction is fully dependent on accurate determination of the positions of atom columns in the 2-D projection images. This is the purpose of this preliminary image preparing procedure.

4.2.1 Step 1a: Geometric Correction of Image Distortions

As discussed in Section 2.3.2, the pixel-by-pixel image recording in STEM mode might render an image of a crystal lattice that appears to be stretched or skewed due to sam-

ple drift during imaging. From knowledge of the theoretical lattice structure for the bulk region of the sample, the effect from the sample drift can be determined and compensated for. A set of coordinates (x_t, y_t) describing the theoretical atom column positions can be used to form a reference lattice describing the "target" of the image correction. Another set of corresponding atom columns at coordinates (x_c, y_c) must be identified in the distorted image. From the difference between these two sets of points, the image distortion is quantified and the distorted image can be corrected. An affine transformation is well-suited for this task, which can be a combination of translation, rotation, scaling and/or shearing. Generally, such transformation can be expressed as a linear mapping $\mathbf{B}$ followed by a translation $\vec{t}$ as follows,

$$\vec{p}' = \mathbf{B}\vec{p} + \vec{t}. \tag{4.1}$$

Here, $\vec{p}$ is the position of a pixel in the distorted image at coordinates (x, y), and $\vec{p}'$ is the resulting position after the transformation, at coordinates (x', y'), while $\mathbf{B}$ and $\vec{t}$ are defined as follows,

$$\mathbf{B} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \qquad \vec{t} = \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}. \tag{4.2}$$

Now, the coefficients $(a_{11}, ..., a_{23})$ can be chosen so that the transformation makes the best possible correction of the image distortion. The goal is for the atom positions in the resulting corrected image to be as similar as possible to the reference lattice. To estimate this optimal affine transformation, the least-squares method is used. A detailed explanation of the implementation of the coefficient estimation and how these relate to scaling, rotation, shear and translation of the image can be found in Appendix A. As a summary, the image distortion correction goes as follows:

1. Identify atom column positions in the distorted image, (x_c, y_c).
2. Create reference lattice from theoretical atom column positions in the bulk, (x_t, y_t).
3. Match the points (x_c, y_c) and (x_t, y_t)
4. Estimate and apply the optimal affine transformation taking atom columns at

(x_c, y_c) to coordinates (x'_c, y'_c) which minimizes the distance to (x_t, y_t) in a least-squares sense.

4.2.1.1 Identifying Atom Column Positions

The first step is to identify the coordinates of N atom columns in the distorted image, (x_c^j, y_c^j) where $j = 1, 2, ..., N$. First, initial peaks are found from local maxima in intensity, with a criterion on minimum pixel distance ensuring that each atom column is only included once. Thereafter, the peak positions are refined by fitting a 2-D Gaussian function to a small area around the peak. An example of initial and resulting peak positions are shown in Figure 4.1, showing the improvement after the position refinement. Further details are given later, in Section 4.2.4.

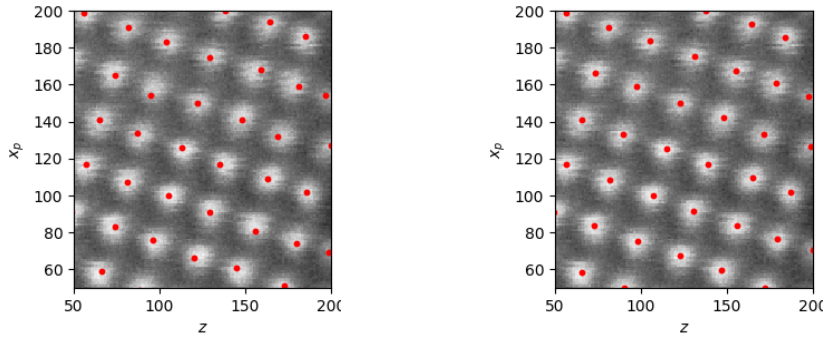


Figure 4.1: Image [001], example of the atom column position determination. **(a)** Initial peaks are found from local maxima in intensity. **(b)** Peak positions are refined by fitting a 2-D Gaussian function to each atom column. Axes in pixels.

4.2.1.2 Create 2-D Reference Lattices

Next, reference lattices with M theoretical atom column positions are generated, with coordinates (x_t^i, y_t^i) where $i = 1, 2, ..., M$. Such reference lattices with an ideal bulk pattern can easily be created from 2-D projections of a set of 3-D points in the same structure as the bulk of the bicrystal being studied. In our example, the YSZ, the cation lattice has an FCC structure (see Section 2.2.2). Figure 4.2 shows the four reference lattices

used for the right side of the GB plane. For the left side, the reference lattices are mirrored about the x_p axis.

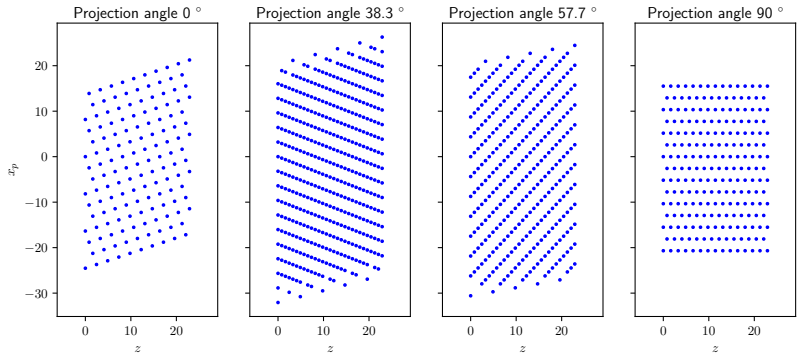


Figure 4.2: Reference lattices representing the theoretical bulk structure of an FCC structure with lattice parameter $a = 5.17$Å, used for estimating the affine transformation for image distortion correction. Both axis are in Ångström (Å).

4.2.1.3 Match Atom Column Positions and Reference Lattice Points

The affine transformation estimation requires the two sets of points to be matched. That is, each point (x_t, y_t) in the reference lattice must be linked to the corresponding atom column position (x_c, y_c) in the experimental image. This is done by first scaling and translating the reference lattice to overlap with the atom columns in the experimental image, see Figure 4.3. Thereafter, for each point in the reference lattice, it is linked to the atom column which is within a distance $d_{min}/2$, if any. d_{min} is the smallest distance between two neighboring points within the reference lattice. The point here is that each column in the reference lattice is matched with maximum one column in the image. Furthermore, not all atom columns are used in the transformation estimation. As an example, in Figure 4.3, the blue dots of the reference lattice that ends up below the image (x_p coordinate less than 0) does not have corresponding atom columns in the image within a distance $d_{min}/2$, and are therefore not included. Furthermore, only atoms at least three atom planes from the GB plane are used to avoid effects from possible lattice distortions in vicinity of the GB.

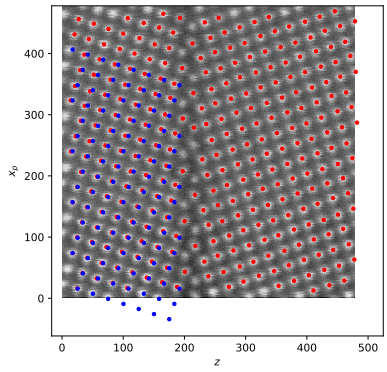


Figure 4.3: Matching between refined atom column positions (red) and reference lattice points (blue). The affine transformation estimation requires the points to be paired together, one red and one blue. Atom columns within 3 atom planes from the GB are ignored. Pairs are made from the rest of the points if the mutual distance is smaller than a criterion d_{min}.

4.2.1.4 Estimate and Apply the Optimal Affine Transformation

The optimal affine transformation minimizes the difference between the transformed atom column coordinates (x'_c, y'_c) and the corresponding reference lattice column coordinates (x_t, y_t). Estimating these coefficients corresponds to minimizing the residuals r_k,

$$R = \sum_k r_k = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left[(x_c'^k - x_t^k)^2 + (y_c'^k - y_t^k)^2\right]}. \tag{4.3}$$

Here, K is the number of atom column pairs identified in the previous step. This corresponds to solving the least-squares problem

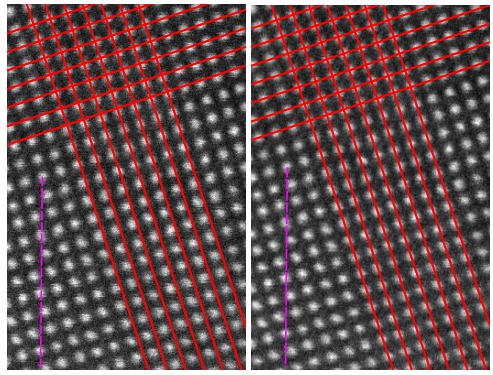
$$\underset{\mathbf{B},\vec{t}}{\arg\min} \left\| \left(\mathbf{B}\vec{p}_c + \vec{t}\right) - \vec{p}_t \right\|^2, \tag{4.4}$$

see Equation (4.1) for an explanation of the symbols.

When preparing the bicrystal sample, the aim is to create a sample that is perfectly symmetric about the GB plane. However, slight deviations in the rotation about the tilt axis (normal to the rotation axis z) can occur. Also, one side might be translated with respect to the other. Because of this, the reference lattice must be fitted to the two sides of the crystal separately. In practice, two affine transformation matrices are estimated,

one for each side. The average of the two is then applied to the full image to correct the distortions.

Figure 4.4 shows Image [001] before and after the averaged affine transformation is applied. Straight lines (atom planes) are preserved, while angles and distances between points are changed. The result is atom columns nicely arranged in a square grid, as is expected in the [001] projection of a crystalline sample with FCC structure.



(a) Original image before affine transformation

(b) After affine transformation

Figure 4.4: Image [001] **(a)** before and **(b)** after affine transformation. The red lines makes it clear how much the atom planes in the uncorrected image deviates from the theoretical square grid. The plane {310}, indicated by a purple dashed line, is vertical in both images. This makes it clear that e.g. rotation alone would not have been sufficient to compensate for distortions from the imaging process. Instead, a combination of rotation, scaling and shearing was needed, combined in an affine transformation.

4.2.2 Step 1b: Scaling

In the matching between image and reference lattice described above, the size of the reference lattice is changed to roughly match the image. Thereafter, the images is scaled to match the reference lattice size, as part of the affine transformation. Since the theoretical distances between atoms in the reference lattice (ideal lattice projection) are known, we can now read the nm-to-pixel ratio of the image directly from this.

4.2.3 Step 1c: Noise Reduction by Image Averaging

One fundamental characteristic of the electron microscope is the unavoidable statistical fluctuations in the number of electrons counted by the detector, so-called shot noise. In Section 2.3.2, arguments are made that this type of noise can be reduced by averaging over several images. In this case, with a highly regular lattice structure with GB units appearing periodically, we can instead average over sections within the same image containing one such unit each. This way, the signal-to-noise ratio (SNR) is greatly enhanced. Another advantage of this approach is that any possible differences between the individual GB units are averaged out. Theoretically, the GB units are identical, but slight deviations might occur in practice. As we are interested in describing the general structure of the Σ5 GB at hand, this image averaging is key to reduce any randomly occurring structural irregularities.

First, the GB unit locations must be identified. This is done by pattern recognition using cross-correlation (similar to [90]). The idea is to copy a small section of the image containing the GB unit, called a *template*, for then to slide it across the image while measuring the "similarity" of the template and the region of the image currently overlapping. Such measure of similarity can be given in terms of the cross-correlation function (CCF), defined as follows,

$$CCF(x,y) = \frac{\sum_{x'}\sum_{y'}\left[i(x',y') - \langle i(x',y')\rangle\right]\cdot\left[t(x'-x,y'-y) - \langle t\rangle\right]}{\sqrt{\sum_{x'}\sum_{y'}\left[i(x',y') - \langle i(x',y')\rangle\right]^2 \sum_{x'}\sum_{y'}\left[t(x'-x,y'-y) - \langle t\rangle\right]^2}} \tag{4.5}$$

Here, $i(x',y')$ is the image and $t(x',y')$ is the template. Angle brackets denotes averaging. Figure 4.5 illustrates the concept for shifting the template along one axis, corresponding to keeping x fixed and let y go from 0 to y_{max} (image size in pixels). Figure 4.6 shows the experimental images before and after averaging.

4.2.4 Step 1d: Identify 2-D Positions of Atom Columns

This part of the algorithm is similar to the atom column identification described for Step 1a (Section 4.2.1.1), i.e. by Gaussian fitting to locating intensity peaks, only now applied to the distortion-corrected and averaged images. Some more details of the al-

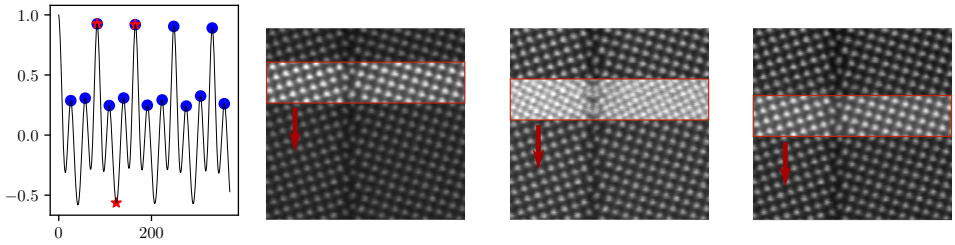


Figure 4.5: Illustration of cross-correlation between the full image [001] and a smaller section of the image (template, in red frame) containing one GB unit. **(b)**, **(c)** and **(d)** show three snapshots from when the template is shifted vertically across the image and the cross-correlation is measured. **(a)** shows the cross-correlation function (CCF) between template and image, calculated with Equation (4.5). Blue circles marks the local maxima indicating a good match between image and template, corresponding to locations of other GB units. Red stars indicates the three positions of the iamge section illustrated in **(b)** local maximum - good match, **(c)** local minimum - not a good match, and **(d)** local maximum - good match.

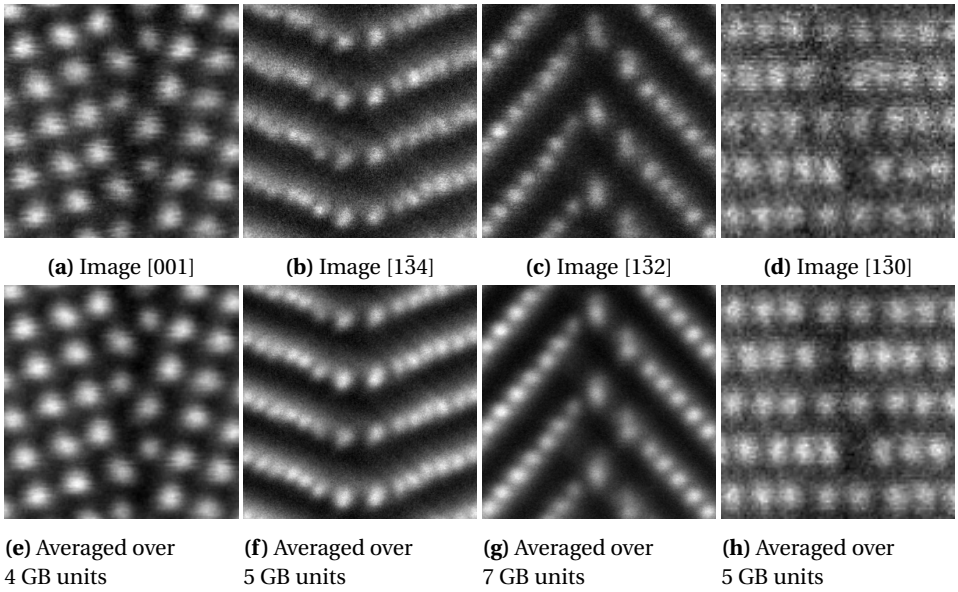


Figure 4.6: Sections of HAADF-STEM images of the YSZ bicrystal, illustrating the effect of image averaging (Step 1c). **(a)-(d)**: Original images. **(e)-(h)**: Noise is reduced by averaging over several grain boundary units.

gorithm are included in the following.

Initial Atom Column Positions: Max Intensity

This part of the code finds all peaks in intensity above a specified level I_l, calculated as

e.g. 80 % of the maximum intensity in the image, at a minimum pixel distance d_{min}. Both parameters are adjusted so that all atom columns are identified. Peaks on the edges of the image are ignored.

Starting with the pixel of maximum intensity, its coordinates (x_1, y_1) are added to the peak list. Thereafter, the pixel and the area within a radius of d_{min} is set to zero. This avoids peaks corresponding to the same atom column appearing in the list several times. The image is searched again and a new pixel is now found as the one with the highest intensity. It is added to the list and set to zero, along with the neighboring pixels. This is repeated until no more pixels are found with an intensity above the specified level I_l.

Refining Atom Column Positions: 2-D Gaussian Fit

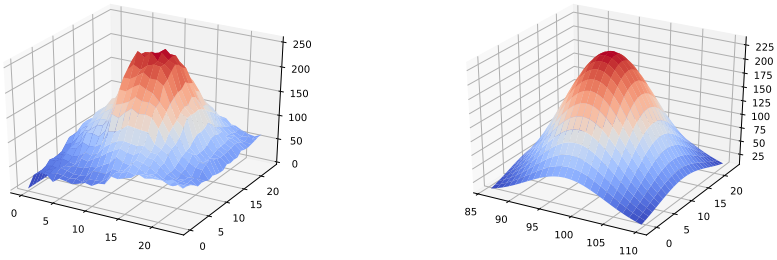
The peak positions are refined by performing 2-D Gaussian fitting to a small area of size $r_g \times r_g$ around each intensity peak, as illustrated for one atom column in Figure 4.7. The Gaussian function is defined as follows,

$$f(x,y) = \frac{A}{2\pi\sigma^2} e^{-\left(\frac{(x-x_0)^2}{2\sigma^2} + \frac{(y-y_0)^2}{2\sigma^2}\right)} \tag{4.6}$$

Here, A is the peak amplitude and σ is the standard deviation, related to the width of the peak, and (x_0, y_0) is the center of the fitted Gaussian function. The parameters A, σ, x_0 and y_0 are estimated in a least-squares sense[91]. The new peak position is set to the center position of the fitted Gaussian function, (x_0, y_0). The process is repeated until positions are stable. The result for all four images is shown in Figure 4.8. This will be the starting point for the 3-D reconstruction.

4.3 Part 2: Iterative Tomographic Reconstruction

The following reconstruction process will not be performed with the full experimental images, as is common for tomographic reconstruction. Instead, only the coordinates of the atom column positions will be used, as determined in the final step of the im-



(a) **(b)**

Figure 4.7: Atom column position refinement by iterative 2-D Gaussian fitting. **(a)** Original intensity distribution in an area of fixed size $r_g \times r_g$ centered at one intensity peak. The pixel of maximum intensity is the initial atom column position. **(b)** The fitted Gaussian function. The center of the Gaussian function is set as the new atom column position. A new Gaussian fitting is done, this time to an area of size $r_g \times r_g$ centered at the new position. This process is repeated until the new position does not deviate much from the old, meaning that a stable position is found.

age preparation part. In practice, these coordinates are an efficient "summary" of the projection images, containing all the information relevant for reconstructing the 3-D structure. This approach exploits the sparse nature of a crystal lattice, greatly reducing the amount of input information to the reconstruction process and thereby reducing an otherwise extensive computational task. This and other advantages that follows the proposed method will be discussed in more detail in Chapter 5. Therefore, in the following, the word "image" refers to the set of 2-D atom column positions identified within each experimental image.

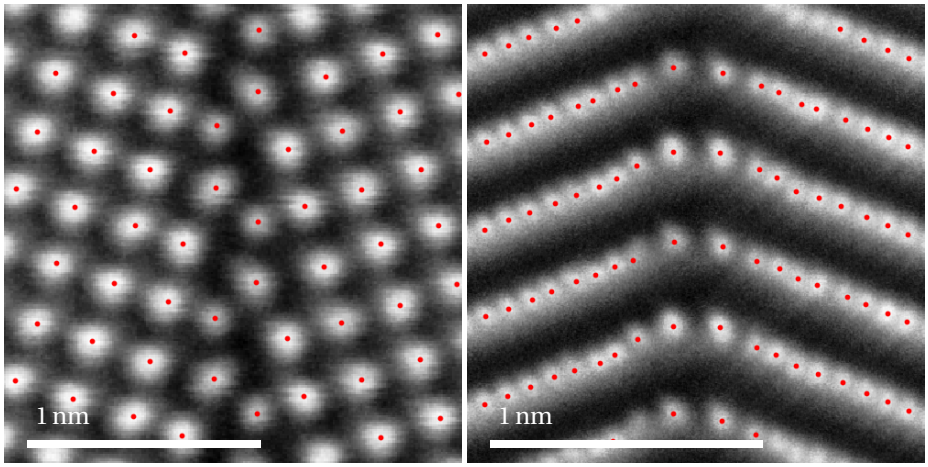
As illustrated in Figure 4.9, the main steps in the 3-D reconstruction algorithm are as follows.

2. **Reconstruction of the 3-D Atomic Structure**

   (a) **Create initial 3-D structure** by direct back-projection

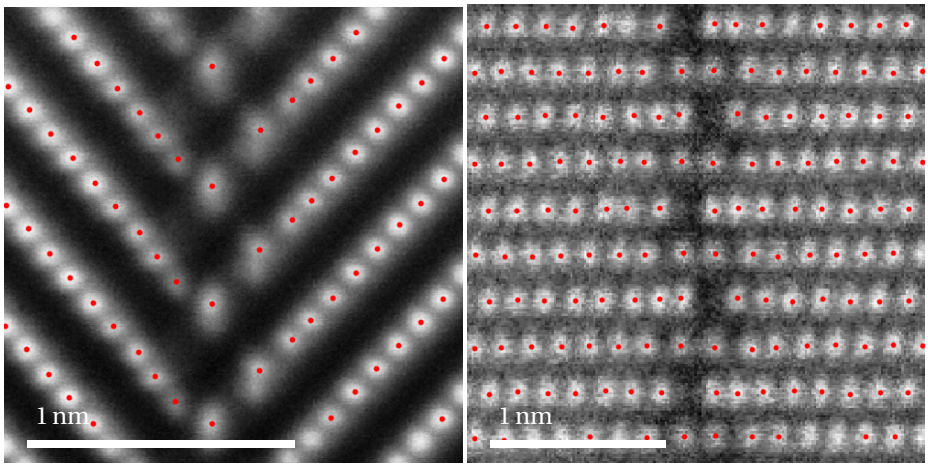
   (b) **Perform alignment refinement** by iterative re-projection and reconstruction

   (c) **Add image** by aligning it to re-projection in a new direction



(a) Image [001]

(b) Image [1$\bar{3}$4]



(c) Image [1$\bar{3}$2]

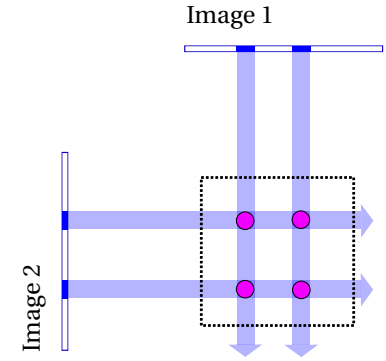
(d) Image [1$\bar{3}$0]

Figure 4.8: Distortion-corrected, scaled and averaged experimental HAADF-STEM images with red dots showing atom column coordinates that will be used in the 3-D structure reconstruction. Each image is 3.4 nm wide, centered about the GB plane. These coordinates are the starting point for the 3-D reconstruction.

(d) **Repeat** Step 2a and Step 2b

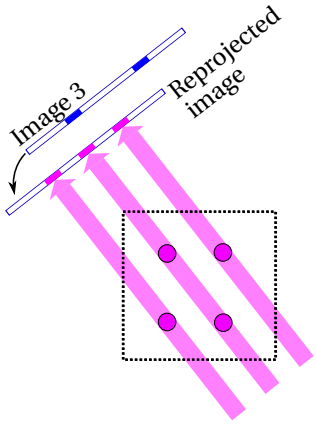
(e) **Repeat** Step 2c and onward to add more images

In brief, the reconstruction is done by introducing one image at a time, and for each new image, an iterative process of image alignment through reconstruction and re-projection is performed repeatedly until a certain criterion is reached. Before describ-



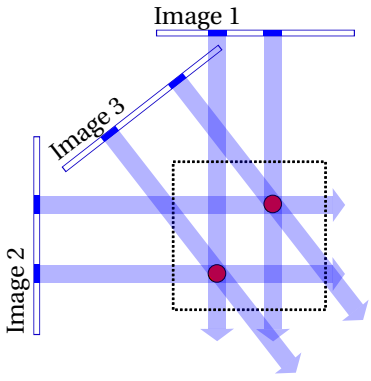
Step 2a:
Initial reconstruction with 2 images. Four potential atoms (pink) is identified in the overlap between the re-projection stripes.

(Step 2b:
Image alignment refinement is illustrated separately in Figure 4.12)



Step 2c:
Re-projection of initial structure in direction 3. Image 3 is aligned to the re-projected image.

(Repeating Step 2b:
Image alignment refinement)



Repeating Step 2a:
Reconstruction with three images. With additional information from image 3, two atoms are removed. The stripes from all three images overlap in only two sites, leaving two potential atoms (red).

Figure 4.9: Algorithm 2 for tomographic reconstruction, illustration for one slice parallel to the GB plane.

ing each step of the algorithm in detail, some general concepts are explained. In the following, the 2-D projection images will be referred to as "image n", where n is an integer 1, 2, 3 etc., referring to the order in which the images are introduced to the reconstruction process. Each of the images are projections of the sample along a projection direction which can be described by the angle θ_n that the sample is rotated about the rotation axis z. Atom columns in image n will have the coordinates (x_{pn}, z) where p stands for *projection*, referring to an axis normal to the projection direction for this image and normal to the rotation axis, see Figure 4.10 for an illustration. A more detailed description is given in the following.

Finding atom positions in 3-D from projections

Consider an atom column in image 1 at coordinates (x_{p1}, z_0). From this information alone, its constituent atoms can be positioned anywhere along a line in the xy-plane described by the following equation,

$$x_{p1} = x\cos\theta_1 + ysin\theta_1. \tag{4.7}$$

This equation similar to Equation (2.17) and the concept of *direct back-propagation* introduced in Section 2.4. In image 2, these atoms will be part of different atom columns. Consider one of these columns at coordinates (x_{p2}, z_0). This gives us a second equation for a line in the xy-plane,

$$x_{p2} = x\cos\theta_2 + ysin\theta_2. \tag{4.8}$$

From this, the intersection of the two lines at (x_i, y_i) can be found. Again, the reader is referred to Figure 4.10. In this intersection, there *could* be an atom, contributing to the intensity in both images 1 and 2. That is, *if* there is an atom contributing to the intensity of both columns, this is the position. Therefore, this pair of coordinates is added to the list of possible 3-D atom positions, along with z_0. Depending on the complexity of the atomic structure of the sample, more images might be needed to confirm of discard this position as an actual atom site.

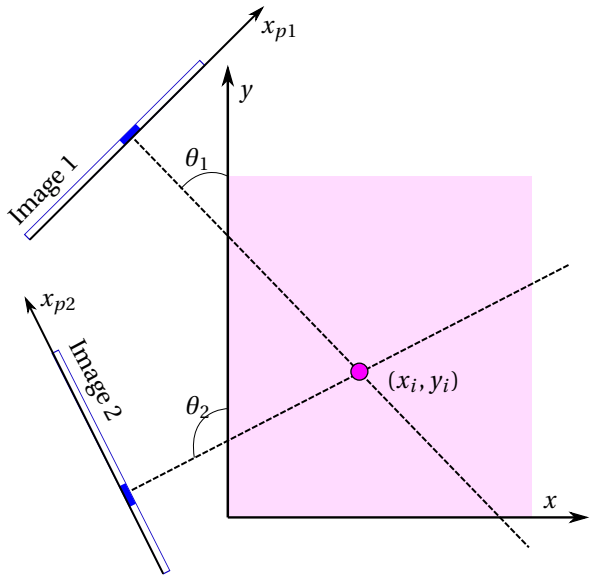


Figure 4.10: Direct back-propagation. If the two stippled lines are within a distance $2r_a$ in the z direction (normal to the paper plane), the lines are considered to overlap. The intersection of the lines is then considered a possible atom location. Symbols are explained in the text.

Taking experimental errors into account

As briefly discussed in Section 2.3.2 and elaborated later in Chapter 5, due to the nature of the imaging technique, HAADF-STEM images might misrepresent the sample structure due to cross-talk and de-channeling effects. In addition, errors that are not sufficiently compensated for in the image preparation part might lead to inaccurate determination of the 2-D atom column positions. To account for all this, a parameter r_a is introduced, called the *atom column radius*, representing the uncertainty of the atom column position (x_p, z). In the implementation of the reconstruction algorithm, all lines within a distance $2r_a$ of each other in the z directions are therefore considered to overlap, and hence representing a possible atom position.

Initial image alignment

In Section 2.4.2, it is concluded that a good image alignment is a central part for a high-accuracy 3-D reconstruction. Some methods for aligning the tomographic tilt series are discussed, including methods based fiducial markers, on cross-correlation between consecutive images (requiring small angle incrementations) or on considering

the center-of-mass of the sample. However, none of these approaches are applicable to the YSZ bicrystal example due to the special sample geometry and the low number of projection images. Instead, the alignment is done iteratively by aligning one image at a time to a re-projection of the reconstructed structure. More details are included later. Firstly, an initial coarse image alignment is needed. It is realized that due to the simple crystal lattice structure with well-spaced atom columns, the initial image alignment only need to be within r_a for the iterative alignment procedure to relax towards the optimal image alignment. In this example, the GB plane can be easily identified in all images, and is therefore chosen as a common starting point.

4.3.1 Step 2a: Reconstruction From Two Images

In general, the N_n atom column positions in image n are listed as pairs of coordinates (x_{pn}^j, z_n^j) where $0 \leq j \leq N_n$. The first step of the reconstruction algorithm considers all atom columns in image 1 and 2.

For all N_1 and N_2 atom columns in image 1 and 2 at positions (x_{p1}^j, z_1^j) and (x_{p2}^j, z_2^j), if the distance along the z axis is less than $2 * r_a$, the intersection (x_i, y_i) in xy-plane is calculated and added to list of possible atom positions. The corresponding z coordinate is set as the mean value of the two image coordinates, $z_i = (z_{1i} + z_{2i})/2$. The list of 3-D coordinates is the initial reconstructed structure.

Now, the question is how to determine the uncertainties r_a in the images. The best way to do this is by trial and error. Since the bulk structure of the crystal is known, the number (or positions) of atoms in the bulk region of the initial reconstruction can be looked at to get an indication of whether the choice of r_a is too high or too low. As an example, Figure 4.11 shows a reconstruction of the YSZ bicrystal with images [001] and [1$\bar{3}$0] viewed from directions [001] and [1$\bar{3}$0], in addition to direction [1$\bar{3}$4]. Figure 4.8 can be used as a reference for the reader of how the bulk structure should look. Figure 4.11a illustrates a low value for r_a at 0.08 Å, where some atoms are missing in the reconstruction. This is not easy to see from the [001] direction, but is clearly visible when the structure is rotated to directions [1$\bar{3}$0] and [1$\bar{3}$4]. In 4.11b, all columns are filled, but

in addition, and a few excess atoms appear. While the extra atoms are difficult to spot in directions [001] and [1$\bar{3}$0] (from which the reconstruction was made), only visible in some slightly widened atom columns. In the [1$\bar{3}$4] direction, however, the extra atoms are easy to spot as they appear between the atom columns rows, in sites that clearly does not agree with an FCC bulk structure. These excess aotms will therefore be easily eliminated in subsequent steps of the reconstruction algorithm, when adding more images to the process. As for the first example with too few atoms in the bulk, however, the empty atom sites will remain empty throughout the reconstruction process. A good rule of thumb is therefore to choose r_a high enough to ensure that all atom sites are filled, on the expense of some extra atoms in other bulk sites.

For this GB geometry, the spacing between bulk atom columns along the z direction is 0.81 Å. If the uncertainty r_a exceeds half of this value, neighboring columns will overlap, making many extra atoms turn up in the reconstruction, as seen in 4.11c. However, due to the low density of atom columns in the GB region, no additional atoms appear here from this increase in r_a. This illustrates the obvious point, that the more dense the structure is, the higher accuracy is needed in the projection images for tomographic reconstruction from a low number of projections to work.

Another important point is that if the GB region is less dense that the bulk, as judged from the projection images, choosing a value of r_a high enough to fill the bulk region will also ensure that all GB atoms are detected. At the other hand, if the opposite is the case, where the GB region appears to have a higher density, a higher value of r_a might be needed.

4.3.2 Step 2b: Refinement of Image Alignment

After the initial reconstruction from two images, the relative alignment of the two images can be adjusted by comparing one image with re-projections of the reconstruction in the same direction as this image is taken from. This image alignment refinement is done in an iterative manner as follows:

1. Re-project the initial structure to one image direction

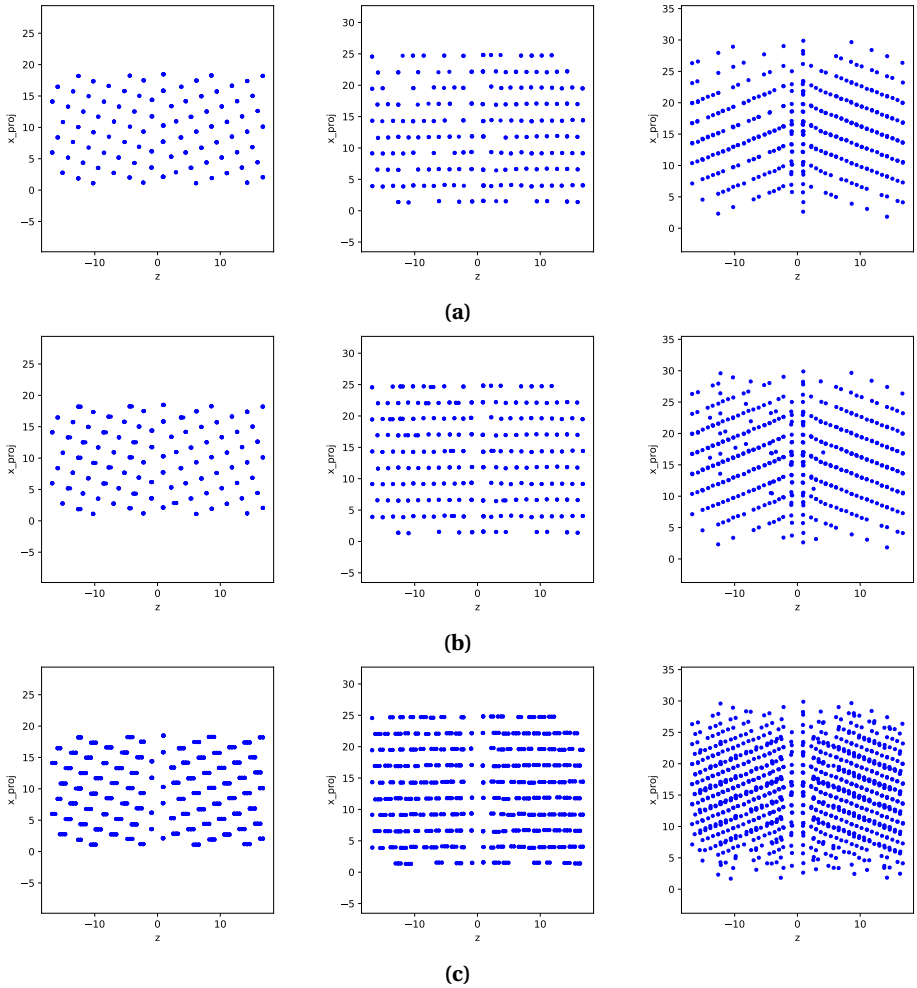


Figure 4.11: Illustration of how the parameter r_a affects the reconstruction result. Each row shows the same structure viewed from directions [001], [1$\bar{3}$0] and [1$\bar{3}$4], from left to right. The values used for r_a are 0.08 Å in **(a)**, 0.3 Å in **(b)** and 0.45 Å in **(c)**. In **(a)**, some atoms are missing in the bulk region, indicating that r_a was chosen too low. In **(b)**, all columns in the bulk are filled. in addition, some columns seem to have doubled in width in directions [001] and [1$\bar{3}$0], appearing as extra atom between rows in direction [1$\bar{3}$4]. Finally, in **(c)**, r_a is chosen higher than half the vertical spacing between bulk atom columns (0.81 Å), making columns overlap and the space between rows in direction [1$\bar{3}$4] nearly filled with excess atoms. Interestingly, even with a high value for r_a and many extra atoms turning up in the bulk, the GB region remains unchanged. This is because of the low density of atoms in this region, so that even with a high level of uncertainty, the lines for each atom column does not overlap with neighboring atom columns.

2. Calculate the center-of-mass (CM) for the re-projection and for the image

3. Align the image to the re-projection so that the CM coincide

4. Perform new reconstruction
5. Repeat until the difference in center-of-mass positions stabilizes on a low level

The first iteration of the refinement process is illustrated in Figure 4.12. After a few iterations, the CM positions of the image and re-projection will not differ much, indicating that the optimal image alignment is found. Note however that in this illustration, the difference between initial image position and re-projection (4.12a) is strongly exaggerated to demonstrate the concept, with a difference in CM positions of 0.2 Å. In practice, the initial shifts are in order 0.01 Å, and reduced to below 1 % of this in three iterations.

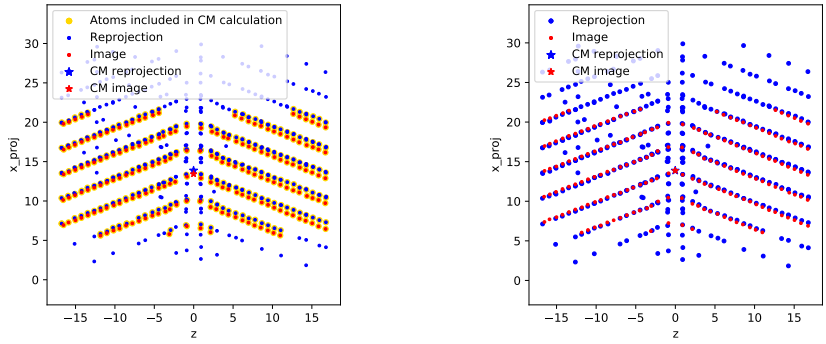
(a) **(b)**

Figure 4.12: Illustration of one iteration in the image alignment refinement procedure, **(a)** before and **(b)** after shifting the image. Red dots are atom column positions in Image [1$\bar{3}$0] with a red star indicating the center-of-mass (CM) of these points. Blue dots (with CM in blue star) are the atom positions in the structure reconstructed from Image [001] and Image [1$\bar{3}$0], re-projected in the [1$\bar{3}$0] direction. Figure **(a)** shows the initial image position. In **(b)**, the image is moved so that the CM positions coincide. With this new image position, a new reconstruction can be made, and thereafter a new re-projection to further adjust the image alignment. After a few iterations, a stable image position is reached, considered as the optimal image alignment. Note that the difference in initial CM positions in **(a)** are exaggerated in this illustration.

4.3.3 Step 2c: Adding a Third Image

Now, to add the next image, the reconstructed structure is re-projected in the direction that this new image was taken from. The alignment of the new image to this re-projection is similar to the CM alignment described above, with one vital difference – the re-projected structure and the new image might contain a different number of atom columns. Therefore, it is important to only take *corresponding* atom columns into account when calculating the re-projection CM, as illustrated in Figure 4.13. In this figure, a reconstruction from Image [001] and Image [1$\bar{3}$0] is re-projected in the [1$\bar{3}$4] direction and plotted with Image [1$\bar{3}$4]. Many more atoms are seen both in the bulk and in the GB region of re-projection. It is important that only atom columns that appear in both

image and reconstruction, marked in yellow in Figure 4.13a, are included when calculating the CM positions, so that the excess atoms in the reconstruction does not affect the alignment. Corresponding atom columns, one in the image and one in the reconstruction, are identified as the nearest neighbor column within some distance d_a, set to e.g. half of the smallest theoretical atom distance in this projection.



(a) **(b)**

Figure 4.13: Illustration Step 2c, adding a new image. Image [1$\bar{3}$4] (red dots) is aligned to the re-projection of the structure reconstructed from Image [001] and Image [1$\bar{3}$0] (blue dots). **(a)** Before alignment. Only corresponding atoms are included in the CM calculation, as indicated in yellow, meaning that excess atoms in the bulk and GB are ignored, along with the part of the reconstruction that is bigger than the image. **(b)** The image is shifted so that its CM (red start) align with that of the re-projection (blue star), giving a much better overlap between image and re-projection.

4.3.4 Step 2d: Reconstruction From Three Images

Tomographic reconstruction with three images is done pair-wise. A list of intersections for lines from atom columns in two and two images are computed, leaving us with three lists of intersections, or *possible atom positions*. A "true" atom would be placed within an atom column in *all three images*, and hence give rise to three intersections at this atom site. See Figure 4.14 for an illustration. As an example, referring again to Figure 4.13, the excess atoms in the GB region in the reconstruction from Image [001] and Image [1$\bar{3}$0] are on the list of intersections from these two images, but would not be on the [001]-[1$\bar{3}$4] or [1$\bar{3}$0]-[1$\bar{3}$4] lists. Thus, these atoms would not turn up in the reconstruction from all three images.

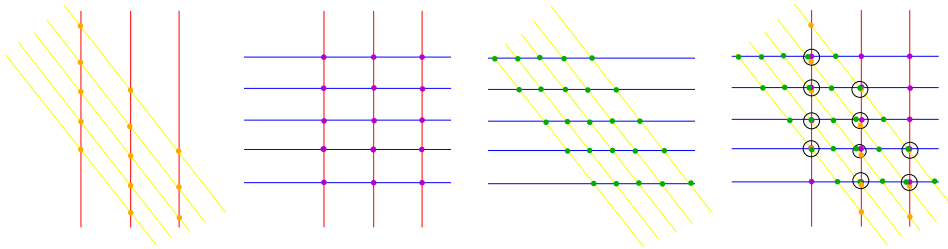


Figure 4.14: Illustration of reconstruction from three images, with the paper plane corresponding to the xy-plane. Lines in one color represents the back-projection from one image. Intersections (marked with dots) from two and two images are computed, as shown in figure **(a)**, **(b)** and **(c)**. In **(d)**, lines from all images and all intersections are plotted together, and sites with intersections from all three image-pairs are marked with a black circle. This circle can illustrate the parameter d_i. If three intersections lies within a circle of this diameter, the site can be considered a possible atom position, see Figure 4.15.

If three intersections are "close enough", as defined by a 3-D distance limit d_i giving room for some error, they are considered as an atom position in the new structure. The atom position coordinates are calculated as the center-of-mass of the three intersections. The parameter d_i is related to the atom column radius r_a, introduced earlier as representing the uncertainty in the 2-D coordinates of the atom positions, as well as the projection angles. This is illustrated in Figure 4.15, showing back-projected lines from image 1, 2 and 3 with atom column radiuses r_{a1}, r_{a2} and r_{a2}. With image 1 and 2 at 0° and 90° and image image 3 at an angle θ, the relation to d_i is as follows,

$$d_i = \frac{r_{a1}}{\sin\theta} + \frac{r_{a2}}{\cos\theta} + \frac{r_{a3}}{\sin\theta\cos\theta}. \tag{4.9}$$

As an example, with image 3 at an angle $\theta = 38.3°$ and atom column radiuses $r_{a1} = r_{a2} = r_{a2} = 0.3\,\text{Å}$ gives $d_i = 1.5\,\text{Å}$. Furthermore, as noticed from Figure 4.14d, the value of d_i can vary a lot without causing any changes to the final structure. This agrees with the discussion above on how to find a suitable value for r_a, concluding that the rules for choosing this value are not very strict. Plotting one layer (in the xy plane) of the structure, similar to what is shown in this figure, can help when investigating how big the variations in intersection positions are, and hence, how big the value of d_i must be to detect all atom positions.

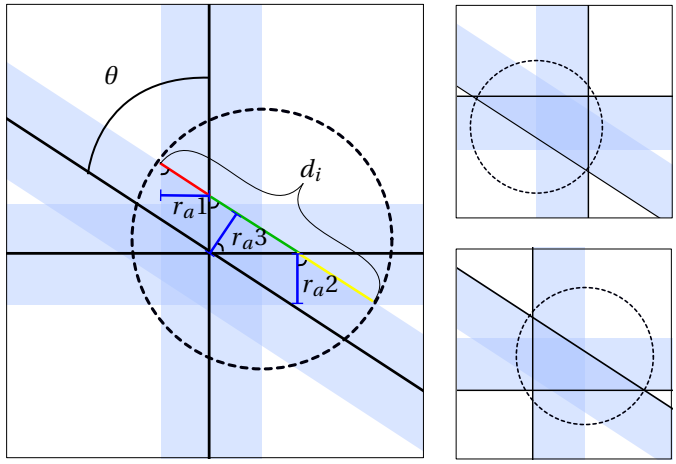


Figure 4.15: Illustration of the geometric relationship between d_i, r_a and projection angles. The three black lines are back-projected lines from images 1, 2 and 3 at angles 0°, θ and 90° with respect to a vertical axis. The blue region centered around each line represents the uncertainty in each atom column position in 2-D, and has a width r_{a1}, r_{a2} and r_{a3} ($r_{a1,2,3}$ for short) for image 1, 2 and 3, respectively. The smaller figures on the right side illustrate the two "worst case scenarios" where the line intersections are at a maximum distance. The dashed circle with diameter d_i shows that the two intersections along the line from image 3 have the biggest distance, as the third intersection lies within this circle. The distance d_i is related to $r_{a1,2,3}$ through Equation (4.9), which is obtained from geometrical considerations of this figure. The tree terms in the equation gives one segment each of the distance d_i, marked in the figure with red, green and yellow. The length of the segments are found from $r_{a1,2,3}$ by realizing that all the angles indicated by a curved line corresponds geometrically to the angle θ.

Next, Step 2b is repeated, refining the alignment of image 3 with respect to the re-projection in direction 3, using center-of-mass of common atoms as described in Section 4.3.2. If more images are to be added, Step 2c, 2d and 2b are repeated for each new image.

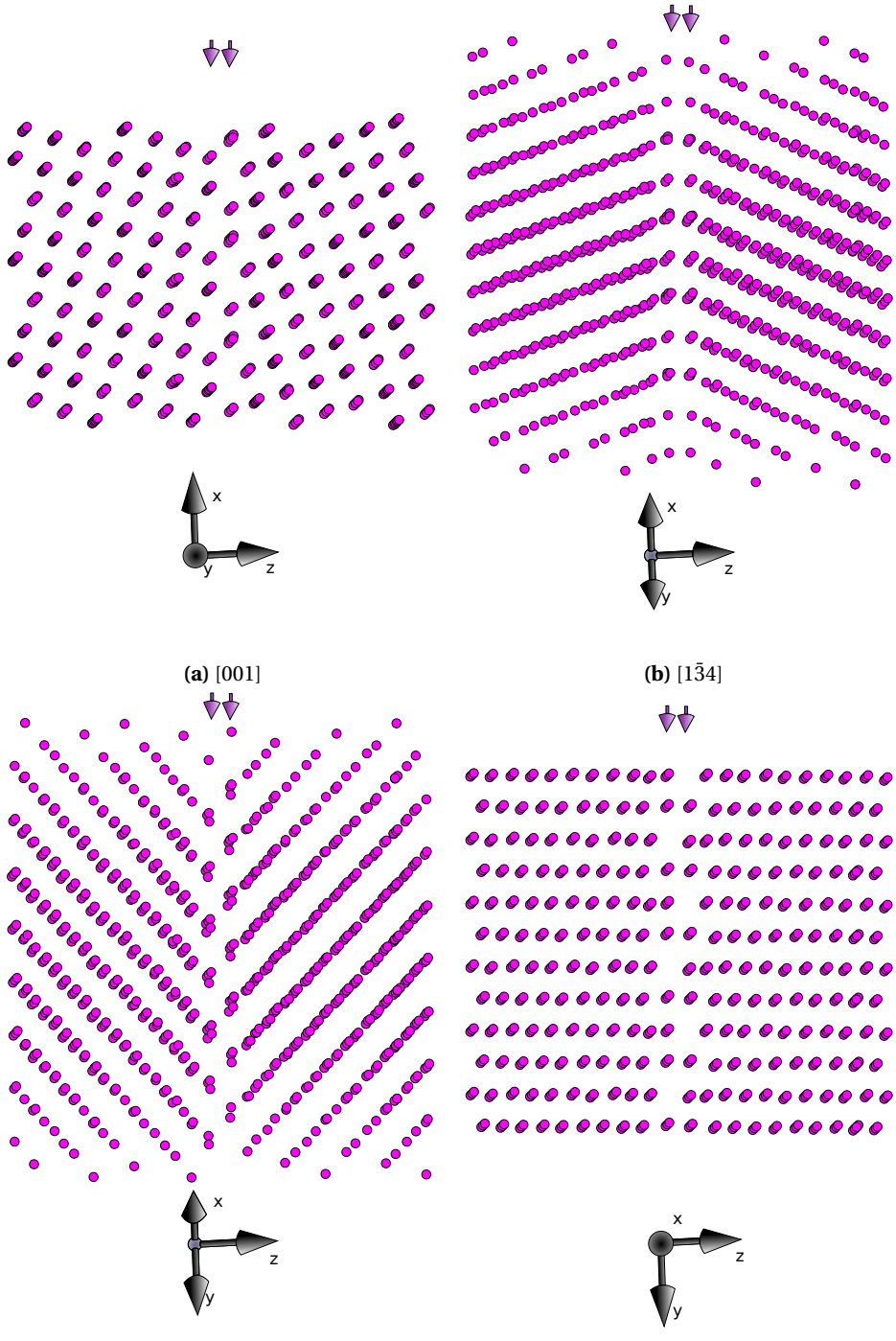
Chapter 5

Results and Discussion

In this chapter, results are presented from the application of the developed reconstruction method to the yttria-stabilized zirconia bicrystal model of a symmetric tilt $\Sigma 5[100]/\{310\}$ grain boundary. Firstly, precision in the reconstruction results are discussed and intensities in the experimental images are looked at. Thereafter, more general comments on the reconstruction method follows, with respect to how many projection images to use, The reconstruction results are discussed in terms of new discoveries related to the material structure of this grain boundary. Thereafter, a more general discussion of the developed method follows, including comments on accuracy, limitations, applicability to other problems and possible improvements.

5.1 3-D Atomic Structure of the YSZ Bicrystal Model

Figure 5.1 shows the result from the 3-D reconstruction of the $\Sigma 5[100]/\{310\}$ grain boundary from HAADF-STEM images. This structure was created using only three of the images, from directions [001], $[1\bar{3}4]$ and $[1\bar{3}0]$ corresponding to rotation angles 0°, 38.3° and 90° about the rotation axis normal to the GB plane (z axis). In Figure 5.2, the reconstructed structure is re-projected in four directions and plotted on top of the corresponding image, showing a very good agreement with the experimental images. The fourth image, Image $[1\bar{3}2]$ at rotation angle 57.7° is included to substantiate the structure of the reconstructed 3-D model.

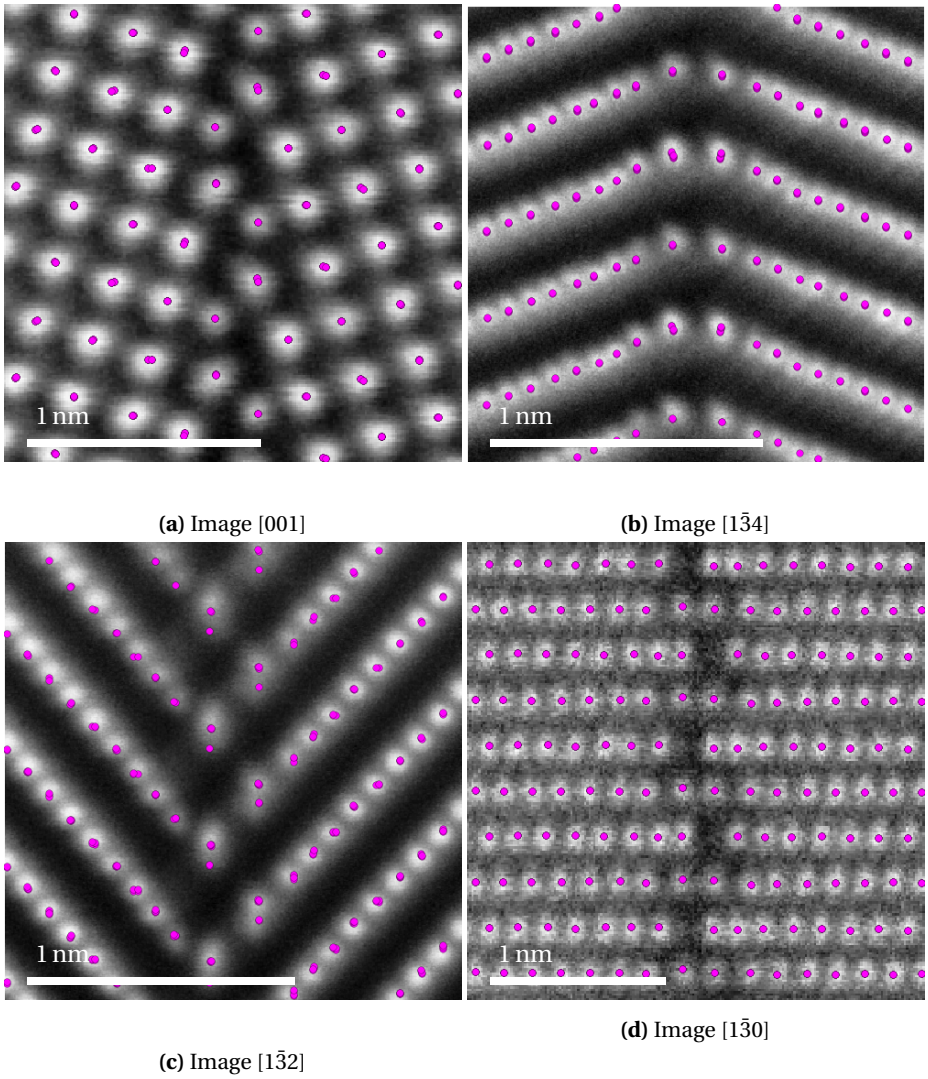


(a) [001] **(b)** [1$\bar{3}$4]

(c) [1$\bar{3}$2] **(d)** [1$\bar{3}$0]

Figure 5.1: The reconstructed structure seen from four directions, 3 by 3 unit cells. The sub-caption indicates the approximate view direction - the structure is further rotated 1° about the rotation axis and vertical axis for better 3-D visualization. The two GB atom layers are indicated with arrows.

The reconstructed structure closely resembles a perfect FCC lattice, but with two atom layers exhibiting a very different atom arrangement. These two layers, indicated by arrows in Figure 5.1, will in the following be referred to as *the GB layers*. A plane in the center between the two GB layers is defined as the *the GB plane*.



(a) Image [001]

(b) Image [1$\bar{3}$4]

(c) Image [1$\bar{3}$2]

(d) Image [1$\bar{3}$0]

Figure 5.2: The result of the 3-D reconstruction from three images. Re-projections of the 3-D structure in four directions are plotted in purple on the corresponding HAADF-STEM image. Good agreement between reconstructed structure and image is seen. The images in **(a)**, **(b)** and **(d)** were used in the reconstruction, while the image in **(c)** was not used.

5.1.1 Comparison with a perfect FCC lattice

The cation lattice in YZS is known to have an FCC structure. In the following, the precision of the reconstructed structure is evaluated in terms of interatomic distances and atom position displacements with a perfect FCC lattice as a reference. However, only the bulk region of the reconstructed structure can be directly compared with an ideal lattice, as the true atom positions in the GB region are not known. Therefore, firstly, a clear definition of the GB region must be found.

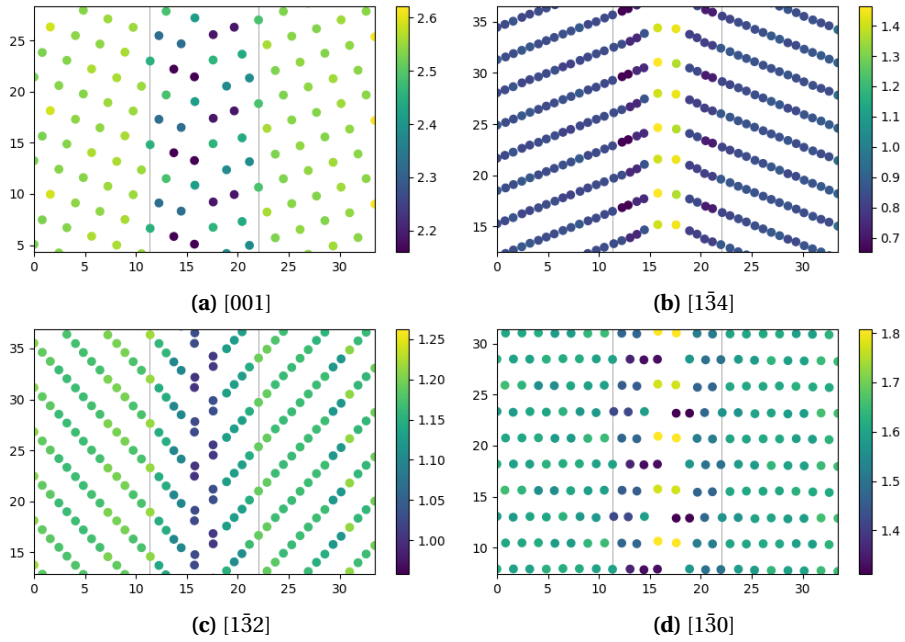
5.1.1.1 Determining the width of the GB region

The structure in the bulk region of the structure closely resembles a perfect FCC lattice, while a slight compression normal to the GB plane is observed in the atom layers in the vicinity of the GB. To visualize this, the structure is plotted viewed from four directions in Figure 5.3, with colors indicating the projected distance to the nearest neighboring (NN) column. Based on the observed structural deviations, the GB *region* is defined to be 10 nm wide, including five layers on each side of the GB plane.

5.1.1.2 Precision

Precision is the statistical spread in repeated measurements. In [92], Bals et. al. define precision in experimental images as the standard deviation (s.d.) in measurements of atom column separations. Using this definition, the precision in the four experimental images after distortion correction and image averaging (Steps 1a and 1c in the image preparation part) is found to be 4–5 pm. The position of each column was determined by Gaussian fit (Step 1d), and the precision estimated from the distances between these positions. Only atom columns in the bulk are included, ignoring 5 atom layers on each side of the GB plane to avoid any contribution form variations in bond lengths in this region.

The same measure of precision can be used for the reconstructed structure in 3-D. Measuring the distance between atom positions in 3-D, the s. d. is 4 pm, agreeing well with the precision measured in the experimental images which were the starting point for



(a) [001] **(b)** $[1\bar{3}4]$

(c) $[1\bar{3}2]$ **(d)** $[1\bar{3}0]$

Figure 5.3: Plots of atom column distances in the reconstructed structure in four directions. Colors indicate projected distance to nearest neighboring (NN) atom column (distance normal to the view direction). Units of axes and colorbar are Ångstrom. The grey lines marks the 6th vertical layer of atoms from the GB plane, at about 5.3 Å on each side of the GB plane. The bulk region outside these lines has a homogeneous color, indicating even spacing between atom columns. The layers between the grey lines and the two GB layers have slightly smaller NN distances, hence the darker color. From this observation, the GB region is determined to be 10 Å wide.

the reconstruction. Again, the GB region is not included in the precision estimation. The mean value of the distance to NN atoms is 3.65 Å. This is in accordance with the theoretical value of a perfect FCC structure with lattice parameter $a = 5.170$ Å, where each atom has 12 NN atoms at a distance 3.656 Å.

Another way to assess precision in the reconstructed structure is to measure the mean value of atom displacements from a perfect FCC lattice. For one side of the GB region at a time, a perfect FCC lattice is fitted to the structure by minimizing the root-mean-square displacement between reference lattice points and atoms (similar to the approach in [58] for displacement estimation in 2-D). The mean value of displacements for atoms from the perfect FCC lattice is found to be 6 pm, slightly higher than the s.d. precision measure of 4 pm. For comparison, Yang et. al. estimates a 3-D precision

of 22 pm for tomographic reconstruction of a 8.4 nm FePt nanoparticle[15]. This reconstruction was done from 68 ADF-STEM projection images using atomic electron tomography (AET).

5.1.2 Atom Column Densities

Figure 5.4 shows a schematic illustration of the atom column densities. The bulk atom columns within one image all have the same number of atoms (density), and are therefore plotted in the same color. In the GB layers, some density variations are seen. A brighter color indicates columns with half the density (Figure 5.4a and 5.4c), while a darker color indicates double density (Figure 5.4b). Furthermore, a thorough analysis of how the atom columns in the different projections are linked gives increased confidence in the reconstructed structure. In Figure 5.4d, the circle with dashed line indicates an "empty" atom column, giving rise to the prominent zig-zag pattern along the GB in Image $[1\bar{3}0]$ (Figure 5.2d). This empty column is found to be the reason for the half-filled columns when observing the structure from the other directions. If the columns marked with a dashed circle in Figures 5.4a, 5.4b and 5.4c was filled to have the same density as the other columns in the GB, this would give fully filled columns in the positions of the dashed circles in Figure 5.4d. However, these columns are without doubt empty, as seen in the experimental image in Figure 5.2d.

The GB structure was further investigated by trying out different atom arrangements. For example, by deleting the half-filled column appearing in Figure 5.4c, columns in other projections disappeared. The same happened when deleting every other atom in the double-filled GB columns in Figure 5.4b. Trying out different cases like this leads to the conclusion that the observed density difference in the GB (half-filled or double-filled columns) has to be true. In fact, the arrangement of atoms in the reconstructed structure is found to be the only configuration matching all projection images. This gives a high confidence in the reconstruction result.

5.1.3 Image Intensities

The intensity profile of the two GB layers are plotted in Figure 5.5, along with the HAADF-STEM image smoothed by Gaussian convolution. As discussed in Chapter 2, although the intensity in a Z-contrast image has a straight-forward relation with the atomic number of the scattering specimen, one should be careful about directly interpreting the intensity near defects like a grain boundary. However, when the 3-D structure and column densities are known, a more detailed discussion is meaningful.

Firstly, a general trend of decreasing intensity closer to the GB is observed in all images in Figure 5.5. According to McGibbon et. al. in [93], reduction in intensity near defects like a GB can be caused by de-channeling for the electron probe, as a result of local strain effects in the defect. One exception from this trend is seen in Figure 5.5b, where every other GB atom column have a strong intensity similar to columns in the bulk. However, as revealed by the reconstruction, the density of these columns are twice as high as in the bulk, explaining why the intensity not lower, despite the prominent de-channeling.

In [81], Findlay et. al. investigates the weakened intensity in the GB region through with image simulations, assuming that strain in the GB region induces distortions in atom positions. Starting out with an ideal structure, the atoms in the GB layers are randomly displaced laterally, in the plane normal to the projection direction. The displacements are normally distributed. With a standard deviation of 0.3 Å, intensities in the experimental images are reproduced. Comments are made that such displacements are larger than what can be expected to be caused by strain alone. Vacancies (missing atoms) are suggested to further reduce the intensity, as indicated in previous reports[94]. Differences in the Debye-Waller factor in different atom columns might also affect the intensity[57].

In both Figures 5.5a and 5.5b, the trend is that every other GB column has a lower intensity, agreeing well with the atom column density as illustrated in Figures 5.4a and 5.4b, respectively. In Figure 5.5c, the profile plots show slightly skewed peaks, and for

the right GB layer, a "shoulder" in the peak is evident, indicating the presence of a second column, although the two columns can not be clearly distinguished. From the 3-D structure, it is found that the projected distance between these two columns (10 pm) is similar to the column distances in the bulk (0.12 nm). However, one column has half the density. This is likely to be the reason why they in the HAADF-STEM image instead appear as one enlarged peak. The total intensity is still weaker than for most bulk columns, but stronger than for the bulk columns nearest to the GB.

In Figure 5.5d, the plot shows a significant difference in HAADF-STEM image intensity between the GB atom columns, although the 3-D reconstruction has revealed that the density of the columns is the same. Looking at the projected distances between atom columns in Figure 5.3d, it is noticed that the atom columns with high intensity has a much smaller distance to the nearest neighbour. This may allow for intensity transfer through the cross-talk phenomena[48], which can explain the difference in intensity in the GB atom columns. A similar observation is made for Figure 5.3a - the distance to nearest neighbor column is smaller for the half-density GB column than for the other GB column. Therefore, intensity contribution from other columns through cross-talk may contribute to even out the intensity differences seen in the profile plot in Figure 5.5a.

5.2 The Reconstruction Method

5.2.1 "Points and lines" approach vs "full image" approach

This approach exploits the sparse nature of the object that is to be reconstructed. Due to the fact that most of the volume in a crystal is empty, the amount of information actually needed to successfully determine the crystal structure very small. Furthermore, because of the highly ordered structure of a crystal, the information needed for the reconstruction can be efficiently summarized in only a few points. By modelling the image intensity as a set of 2-D Gaussian functions, one for each atom column, the image can be described by the parameters of these Gaussian functions (see Section 2.3.2 and Equation (2.14)). Furthermore, as we in this method only need to consider the *position*

of the atom columns in 2-D, the coordinates of the peak centers (μ_{x_m}, μ_{y_m}) are sufficient, leaving us with a set of M pairs of coordinates of M atom columns summarizing the HAADF-STEM projection image.

This approach has several advantages. Firstly, the computation time is reduced. Intensity values in $L \times M$ pixels are converted to a list of N pairs of coordinates needed to describe the positions of N atom columns, greatly reducing the amount of information considered from each projection image. As an example, direct back-projection with two images of size $H \times H$ pixels requires H^4 computations, adding the intensity of each pixel to corresponding voxels in the 3-D reconstruction volume[66]. Thereafter, the reconstruction volume of e.g. $H \times H \times H$ voxels must be searched and possible atoms identified from intensity peaks in 3-D. In contrast, for the method considering only M pair of coordinates from each image, the only calculations needed are a quick reach for which coordinates are sufficiently close in the z direction, before intersections are calculated for two and two lines fulfilling this requirement. These intersections are all possible atom positions (see Section 4.3).

Another important advantage is that no discretization of the reconstruction volume is needed, removing one main source of error in conventional tomography[66]. The reconstruction volume consist of a discrete set of voxels, and each pixel in a projection image is linked to a ray of voxels. Several methods exist for calculating how to weigh the intensity from different pixels in different voxels, but in any case, errors are inevitably introduced from approximations that must be made due to the discrete nature of the problem . For the approach with points and lines instead of images, however, the possible atom position coordinates (line intersections) are not locked to at fixed grid. Furthermore, the accuracy in the calculated position can be more directly accessed through the uncertainty in the 2-D atom column positions in each image.

5.2.2 How many images are needed?

The number of images required for a successful reconstruction depends on the complexity of the structure studied. For the YZS example used in this thesis, three images

were enough to fully determine all atom positions with high confidence. For the FCC bulk region, as little as two images were needed. It was however of importance *which* images were used - images [001] and [1$\bar{3}$0] successfully reproduced the FCC bulk, while combinations including [1$\bar{3}$4] or [1$\bar{3}$2] failed. This demonstrates that not only the number of projection images, but also the projection angles are is of importance.

Figure 5.6 shows the input images (first row) along with three structures (A, B and C) reconstructed from different combinations of input images. The second row in the figure shows structure A, reconstructed from images [001] and [1$\bar{3}$0], demonstrating that the bulk region is reproduced while the GB layers contains too many atoms. This is evident when viewing the structure from directions [1$\bar{3}$4] and [1$\bar{3}$2]. This makes it clear that more image are needed in the reconstruction process. The third row in the figure shows structure B, reconstructed from images [001], [1$\bar{3}$0] and [1$\bar{3}$4]. This is the structure studied in more detail earlier in this chapter. The final structure C is reconstructed from images [001], [1$\bar{3}$0] and [1$\bar{3}$2]. This structure displays a slightly poorer agreement with the experimental images compared to structure B, as seen in the GB atom columns indicated with yellow circles. The biggest difference between B and C is however seen in the columns circled in red. While the number of encircled atoms are the same in B and C, they are shifted so that they appear as two separate columns in B, while merged into one column in C.

The reason for this is seen in the input image (green frame) used in the reconstruction of C, but not in B. In the experimental image (Image [1$\bar{3}$2]), what appears as elongated atom columns in the GB was interpreted as single columns. In discussions earlier in this chapter, arguments were made that this is in reality two separate columns, appearing as one due to the nature of the imaging technique. Now, another strong indication towards the single column being an image artefact is found, seeing that this interpretation leads to poorer agreement with the experimental images. Hence, we can conclude that structure B is the true representation of the YSZ bicrystal sample.

Using a minimum number of experimental images is an advantage in many ways. Acquiring a good-quality sub-angstrom HAADF-STEM image is a challenging task, even

with state-of-the-art microscopes available. In addition, both imaging and sample preparation are a time-consuming processes. Reconstruction from as few images as possible is therefore very welcomed. With the developed method, the experimental images can be acquired and added to the reconstruction process one by one. An initial structure can be made and revised, before concluding whether another image is necessary or not. In contrast, in conventional tomography, a high number of images is usually desired to increase the resolution and accuracy of the reconstruction[66]. When dealing with HAADF-STEM images of crystal lattices, however, as illustrated in this section, adding more images could lead to a worse reconstruction result if the experimental images are misrepresenting the sample structure.

5.2.3 Limitations

It is important to note that the result of the reconstruction is an *averaged* structure, assuming all structure units are identical. This assumption is what allows for reconstruction from images acquired from what in practice is four different samples. The assumption is further applied in the image preparation part when averaging the images over several structure units to improve the SNR ratio and hence the image precision. Furthermore, the imaging technique itself provides averaged images, as the HAADF-STEM image is a projection of the sample. Atoms are is other words not observed individually. An intensity peak in the experimental image is the accumulated signals from all atoms within one column, which in our 30 nm thick YSZ samples corresponds to 30-60 atoms depending on the view direction. Hence, the resulting atom column positions used in the reconstruction are average positions, failing to describe any slight displacements by individual atoms. Moreover, vacancies (empty atom sites) are not detected. In summary, any structural irregularities with periodicity larger than one structure unit will not be accurately represented in the reconstruction result.

5.2.4 Possible Improvements

- **Improve experimental image precision by non-rigid registration.**

  The precision in the experimental images were measured to 4–5 pm, the exact

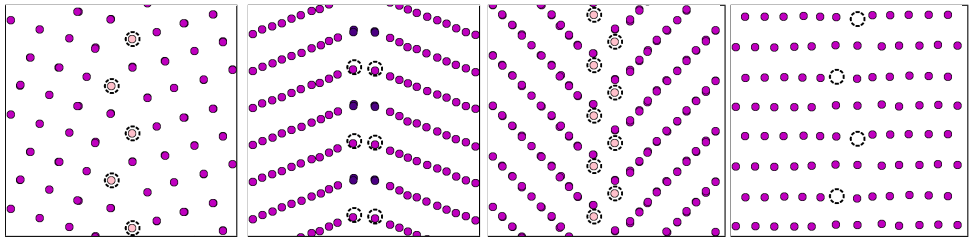
same level as what was measured in ADF-STEM images in [95, 96], obtained by image averaging. Several experimental images were aquired and matched by cross-correlation, similar to the technique used for our experimental technique. However, in [58], Yankovich et. al. reported to reach sub-picometer precision in HAADF-STEM images with a method based on non-rigid registration[97, 63, 98]. This method compensates for image distortions during imaging, increasing the SNR ratio which is usually the limiting factor for the precision. To compare, the rigid registration (our method) applied to the same images resulted in 4–5 pm precision, similar to our result. It is therefore likely that to apply non-rigid registration would improve the precision in our experimental images.

- **Improvements in 2-D atom column identification.**
  Both before and after image distortion correction and image averaging, atom column positions are determined by fitting a 2-D Gaussian function to intensity peaks. However, in the current version of the code, the Gaussian functions are fitted one-by-one to a fixed set of pixels around each atom column. This way, intensity from neighbouring columns might affect the position determination. Ideally, all Gaussians should be fitted simultaneously, using e.g. Mixture Models[99]. This should be easy to incorporate to the peak-finding algorithm as implementations are available in e.g. Scikit-learn[100], see [1].

- **Include functionality for optimizing the uncertainty parameter r_a automatically.**
  In the reconstruction algorithm, the parameter r_a (and thereby d_i) accounts for uncertainty in the 2-D atom column position in the input image. A suitable value is found by visual inspection of the reconstructed structure to check whether atoms were missing in the bulk region. This can be automated by comparing the reconstruction to a reference lattice in 3-D directly.Moreover, in the current version of the code, the same value for r_a is used for all atom columns. Allowing for bigger uncertainty in user-specified columns makes the method more flexible in coping with errors from the experimental images. While the result would be

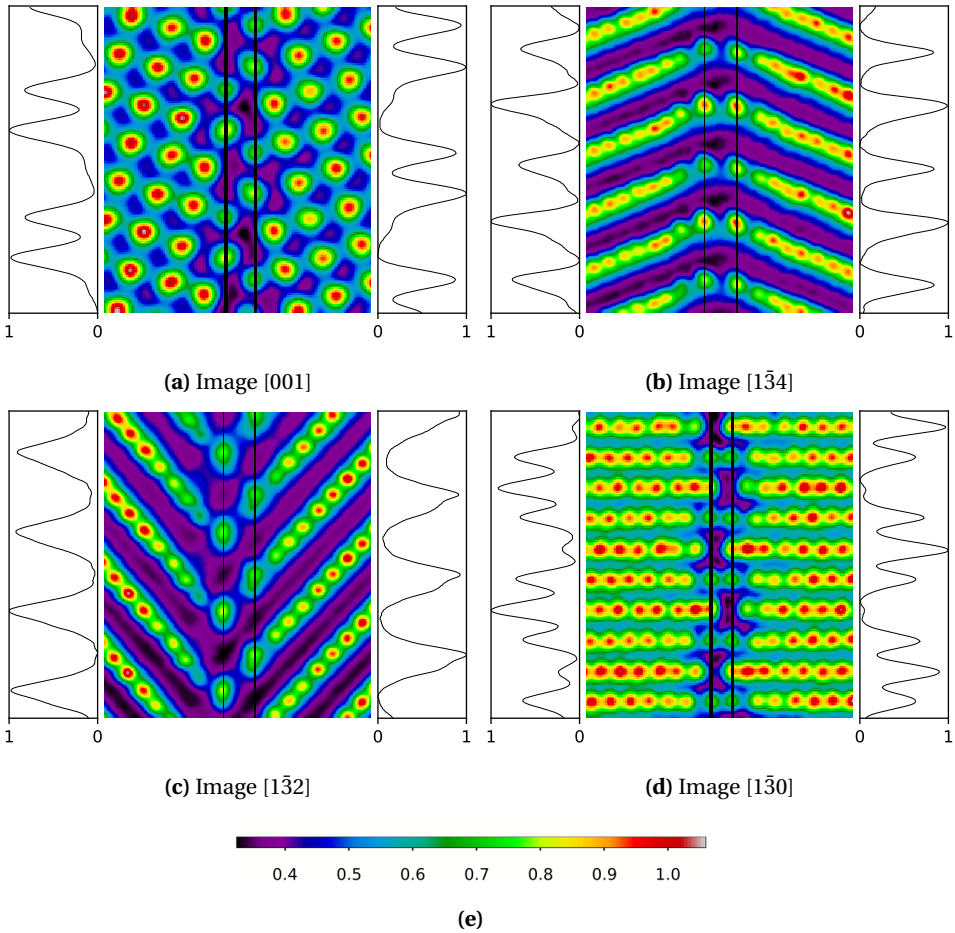
[1] http://scikit-learn.org/stable/modules/mixture.html

identical, these two improvements would simplify the reconstruction process as less manual input would be needed.



(a) Image [001] **(b)** Image $[1\bar{3}4]$ **(c)** Image $[1\bar{3}2]$ **(d)** Image $[1\bar{3}0]$

Figure 5.4: Atom column densities. While all atom columns in the bulk have the same density (pink color), some variations are seen in the GB layers. Brighter colored columns have half the density ((**a)** and **(c)**), darker colored columns have double density ((**b)**). If the columns marked with a dashed circle in **(a)**-**(c)** was filled to have the same density as the other columns in the GB, this would give a fully filled column in the position of the dashed circle in **(d)**. This column is however clearly empty, as seen in Figure 5.2d. From this and other considerations it is concluded that the reconstructed structure is the only atom arrangement that matches all projection images.



(a) Image [001]

(b) Image [1$\bar{3}$4]

(c) Image [1$\bar{3}$2]

(d) Image [1$\bar{3}$0]

(e)

Figure 5.5: Gaussian convoluted experimental images with profile plots of the intensity in the GB layers. Black lines indicate where the profile plot is made from, with the left line corresponding to the profile plot to the left of the image etc. The intensity roughly follows a trend with higher intensity in higher-density columns, see Figure 5.4 for a schematic density plot. Horizontal axes of the profile plots show the relative intensity along the line. The color bar in **(d)** indicates relative intensities for all images.

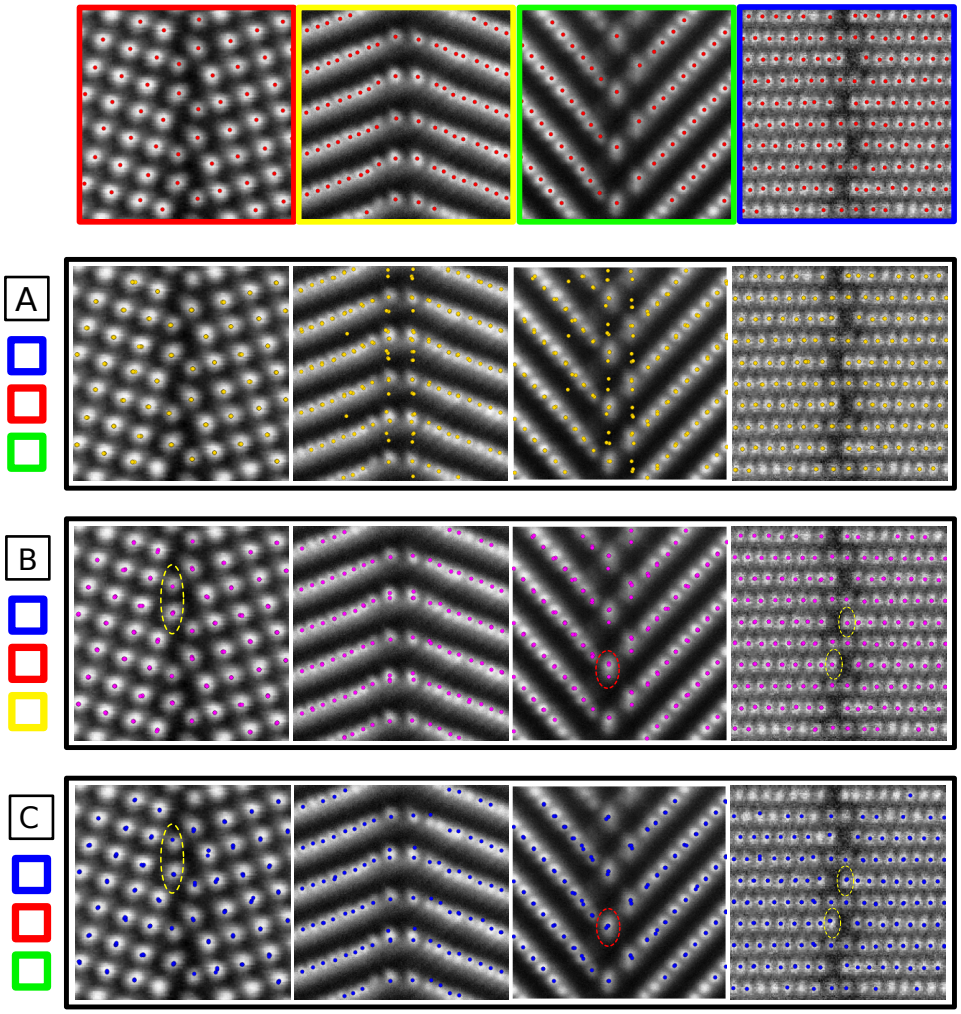


Figure 5.6: Input images (first row), together with three structures reconstructed from different combinations of input images. Each row of images in a black frame, marked A, B and C, is one structure viewed from four directions. The colored squares to the left indicated which of the input images were used in the reconstruction of each structure. In structure A (from input images in red and blue frames), excess atoms are seen in the GB region, clearly not matching with the experimental images in yellow and green frames. Red circles indicate the biggest difference between structure B and C, stemming from the input image in green frame. Yellow circles indicate sites where structure B exhibit a better match with experimental images than structure C.

Chapter 6

Conclusion

In this thesis, a new method for 3-D reconstruction at atomic resolution for crystal grain boundaries has been presented, based on tomographic reconstruction from very few HAADF-STEM images. The method was applied to a bicrystal model of a symmetric tilt $\Sigma 5\{310\}/[001]$ grain boundary in yttria-stabilized zirconia. Through this work, it is demonstrated that by using the atom column positions in images from only three directions, it is possible to accurately determine the 3-D position of all atoms in the grain boundary structure, revealing structural information that is not apparent in the 2-D projection data alone.

Code is written to take care of every step in the process of reconstruction, from raw experimental images to the final reconstructed 3-D structure with all atom positions determined. How to use the code is illustrated in a tutorial included in the appendices. In Chapter 4 each step of the process is explained in detail. In the first part, techniques for preparing the experimental images are presented. Distortion from sample drift during imaging is corrected by applying an affine transformation to the image, estimated from knowledge on the ideal FCC bulk structure. Noise is reduced by averaging the images over several grain boundary units. Finally, the atom column positions are determined by fitting a 2-D Gaussian function to each intensity peak in the image. Thereafter, the algorithm for finding 3-D atom positions is explained. Due to the highly regular structure of a crystal, in combination with the fact that most of the volume is empty space,

the amount of information needed to reconstruct the structure is very small. This is exploited in the new approach applied in this method, where instead of using the full experimental images, only the 2-D coordinates of atom column are used in the reconstruction process. Compared to conventional tomography, one big advantage is that no discretization of the reconstruction volume is needed, removing one source of error. In addition, the computation time is reduced and the image alignment process simplified.

The final reconstructed $\Sigma 5\{310\}/[001]$ GB model was obtained from images in three high-symmetry directions [001], $[1\bar{3}4]$ and $[1\bar{3}0]$, corresponding to rotation angles 0°, 38.3° and 90°. A 3-D precision of 4 pm was estimated by comparing the bulk of the reconstruction with an ideal FCC lattice, the same level as for the experimental input images.

A detailed study of the different GB atom sites lead to high confidence in the reconstructed structure. The atom arrangement in the reconstruction is found to be only possible configuration matching all four projection images. Furthermore, interesting observations were done when closely examining HAADF-STEM image intensities. Cross-talk and de-channeling effects are clearly present, as atom columns with the same density appeared with big differences in intensity, comparable to the differences found between atom columns where one had half the density of the other.

Reconstruction from few images is a great advantage as HAADF-STEM imaging is a challenging task as well as a time-consuming procedure. With this method, only two images were sufficient to successfully reconstruct the FCC lattice of the YSZ bulk, while three images were needed to determine all atom positions in the GB region. The reconstruction result provided new insight to the atomic structure of a $\Sigma 5$ GB, as well demonstrating that tomography in combination with HAADF-STEM imaging holds great promise for future GB structure characterization.

Chapter 7

Further Work

The developed method should be applicable to all grain boundary structures that can be directly imaged at atomic resolution from several directions. For HAADF-STEM, pure tilt grain boundaries are a good choice, but other coincidence-site lattice (CSL) GBs could also be good candidates. Therefore, the natural first step in further work would be to use the method to reconstruct other grain boundary structures.

In Chapter 5, some qualitative discussion was made on the intensity in the HAADF-STEM images. A more in-depth analysis of the image intensity in the GB region could further be done. With detailed knowledge about the 3-D structure, effects like de-channeling and cross-talk can be studied, e.g. through image simulations[94], possibly leading to a better understanding of the image formation in HAADF-STEM and the imaging of defects.

Figure 7.1 shows the EDX mapping of Y in the YSZ bicrystal with projections of the reconstructed structure plotted on top. The color of the atom columns indicates the concentration, extracted by using the reconstructed structure as a reference to determine the precise location of atom columns within each image. While the initial concentration of Y in the bulk was about 19 %, this EDX mapping shows concentrations up to 55 %, clearly demonstrating that grain boundary segregation (GBS)has occurred. In further work, it would be interesting to map the concentration information to 3-D, identify-

ing preferential atom sites for GBS. An approach with discrete tomography (DT) could be tried. The grain boundary phenomena could further be studied through theoretical simulations using e.g. MC or MD. This has previously been done for ideal (CSL) lattices with a simple geometry[44, 42, 22]. However, with the reconstructed structure as an initial structure in the simulation, the results could be more directly compared to the concentrations determined by EDX. Figure 7.2 shows the reconstructed structure expanded to contain two grain boundary planes in order to establish periodic boundary conditions. Oxygen atoms are added so that the ratio to cations is 2:1.

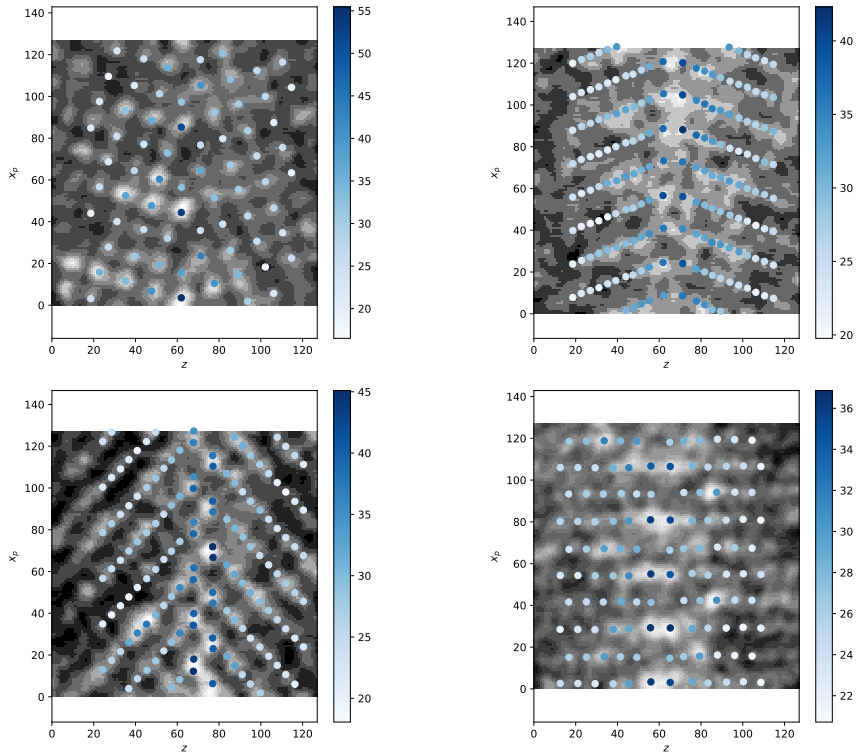


Figure 7.1: Yttrium concentrations in the YSZ bicrystal extracted from EDX images, showing that Y atoms segregate preferentially to specific GB sites. In further work, it would be interesting to use the concentrations to identify Y atom positions in 3-D, revealing information about the grain boundary segregation phenomena in YSZ.

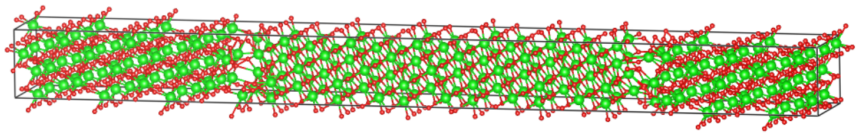


Figure 7.2: The reconstructed structure of YSZ to be used in simulation studies of GBS. Screenshot from visualization in VESTA.

Appendix A

Affine Transformation for Image Distortion Correction

As discussed in Section 2.3.2, sample drift during STEM imaging can lead to a distorted image. In Section 4.2.1, an affine transformation is applied to correct this distortion. Such transformations preserve straight lines, parallel lines remain parallel, while angles between lines and distances between points might change. In brief, the least-squares method is used to estimate the transformation giving the best fit between a set of points in the transformed image and points in a template generated from the theoretical bulk parameters of the sample.

Using homogeneous coordinates, an affine transformation relating a pair of 2-D coordinates (x, y) with another set (x', y') can be expressed as a matrix multiplication as follows[101],

$$\vec{v}' = \mathbf{A}\vec{v}, \tag{A.1}$$

with

$$\vec{v}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \quad \vec{v} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

The transformation can be a combination of any of the following operations, illustrated

in Figure A.1,

$$\text{Pure translation:} \quad \mathbf{A} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.3a}$$

$$\text{Pure scaling:} \quad \mathbf{A} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.3b}$$

$$\text{Pure rotation:} \quad \mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.3c}$$

$$\text{Pure shear, } x\text{-direction:} \quad \mathbf{A} = \begin{bmatrix} 1 & \tan\phi_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.3d}$$

$$\text{Pure shear, } y\text{-direction:} \quad \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ \tan\phi_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.3e}$$

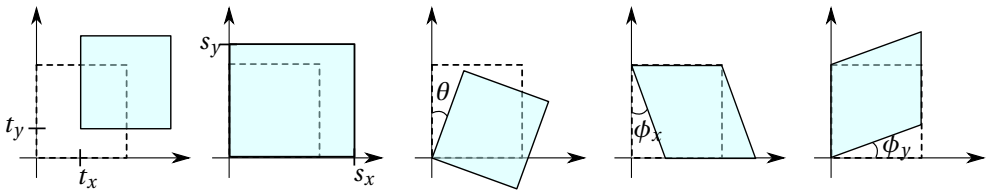


Figure A.1: Illustration of the affine transformations in Equations (A.3)(a)-(e), from left to right illustrating translation, scaling, rotation and shear in x and y directions.

A.1 Implementation

The image distortion correction is based on the affine transformation estimation as implemented in *scikit-image*[102]. This is an open source image processing library for the

Python programming language. The 2-D affine transformation in the class `skimage.transform.Affine` is defined as follows,

$$x' = a_{11} * x + a_{12} * y + a_{13} = s_x x \cos\theta - s_y y \sin(\theta + \phi) + t_x \tag{A.4}$$

$$y' = a_{21} * x + a_{22} * y + a_{23} = s_x x \sin\theta + s_y y \cos(\theta + \phi) + t_y \tag{A.5}$$

Here, s_x and s_y are scale coefficients in the x and y directions and t_x and t_y are components of the translation vector. The rotation and shear angle is denoted θ and ϕ, respectively. Only shear in the x direction is included. From these equations, the effect of the transformation can be calculated from of the matrix coefficients as follows,

$$t_x = a_{13}, \qquad t_y = a_{23}, \tag{A.6}$$

$$S_x = \sqrt{a_{11}^2 + a_{21}^2}, \qquad S_y = \frac{-a_{12}}{\sin\left(\arctan\left(\frac{-a_{12}}{b_{22}}\right)\right)}, \tag{A.7}$$

$$\theta = \arctan\left(\frac{a_{21}}{a_{11}}\right), \qquad \phi = -\arctan\left(\frac{a_{12}}{a_{22}}\right) - \arctan\left(\frac{a_{21}}{a_{11}}\right). \tag{A.8}$$

Least Squares Method

In general, since the affine transformation is defined by 6 constants, it is possible to determine the transformation from 6 points in total, that is, any three points in the input image and the corresponding three points in the output image. In practice, for image corrections, it is however common that many more points are measured, and e.g. the least-squares method can be used to find the best fitting transformation. The details of this method are explained in the following.

Since the coefficients in Equations (A.4) appears linearly, the problem can be re-written as

$$\mathbf{B}\vec{\beta} = 0, \tag{A.9}$$

[1] http://scikit-image.org/docs/dev/api/skimage.transform.html#skimage.transform.AffineTransform

where

$$\mathbf{B} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -y' \\ 0 & 0 & 0 & x & y & 1 & -x', \end{bmatrix}$$

$$\beta^{\mathsf{T}} = \left[a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, c_n\right]$$

The transformation matrix coefficients are found by calculating the coefficients giving the best fitted transformation between two set of N 2-D coordinate pairs (x_i, y_i) and (x'_i, y'_i) where $i = 1, 2, ..., N$. In the *skimage* implementation, this is done by solving a *total* least-squares problem[103], allowing for errors both in the template and image coordinates. This corresponds to minimizing the 2-norm $\left\|\mathbf{B}\vec{\beta}\right\|^2$ under the constraint that $\left|\vec{\beta}\right| = 1$. This is ensured by the normalizing coefficient c_n. For clarity, written in terms of the transformation matrix $\mathbf{A}$, the total least-squares problem can be written as

$$\arg\min_{\mathbf{A}} \left\| \frac{1}{c_n}\mathbf{A}x - x' \right\|^2. \tag{A.10}$$

Appendix B

Python Package: Grain Boundary Reconstruction

This appendix provides an overview of the implementation of methods for 3-D reconstruction of a crystal grain boundary from atomic resolution HAADF-STEM images. A tutorial is provided on how to use the package `grain_boundary_reconstruction` for image preparation. The steps described in Chapter 4 are followed. Again, images from the bicrystal model of a $\Sigma 5\{310\}/[001]$ GB is used to demonstrate the process. The code is included in Appendix C as well as in the zip-file attatched to this thesis containing the full Python package.

B.1 Code Organization

All code included in Appendix C are function definitions, which can be accessed by loading them into a script. As the reconstruction process requires manual input in all steps, it is highly recommended to use an IDE (interactive development environment) where code can be executed section-by-section, e.g. Spyder[1] or Jupyter Notebook[2]. Depending on their main area of use, functions are grouped into *modules*. This is the Pythonic word for a file where function definitions are gathered and saved for later use.

[1] https://github.com/spyder-ide
[2] http://jupyter.org/

When the module is imported in the beginning of a script, all its constituent functions are made available.

All module files are placed within a folder named `grain_boundary_reconstruction` together with an `__init__.py`-file. This makes the Python interpret the folder as a *package*, containing several modules, scrip files and test data.

In the subfolder named `test_data_folder`, original HAADF-STEM images from the YSZ bicrystal example are found. In addition, images are saved here from the different steps in the preparation process. Atom column coordinated are also stored. All files in the folder can be retrieved by functions in the `test_data` module. In this tutorial, images will be retrieved from here for each new step in the image preparation part. By doing so, the reader can try out each step separately and in any order.

Package:
`grain_boundary_reconstruction`

Modules (with link to code in Appendix C):

- `tools` (C.0.0.1)
  Functions for plotting, read and write files etc.
- `test_data` (C.0.0.2)
  Functions for retrieving files in `test_data_folder`.
- `peakFinder` (C.0.0.3)
  Functions for identifying intensity maxima and Gaussian fitting.
- `geometric_correction` (C.0.0.4)
  Functions related to image distortion correction.
- `noise_reduction` (C.0.0.5)
  Functions for creating templates, cross-correlation, averaging.
- `reconstruction` (C.0.0.6)
  Functions for finding 3-D atom positions and image alignment

Subfolders in `test_data_folder`:

- `original_images_sigma5`
- `original_images_atompos`
- `affined_images`
- `averaged_images`
- `averaged_images_atompos`
- `averaged_images_atompos_scaled`
- `ideal_3D_structure`
- `rec_structures`

B.2 Tutorial: Image Preparation

The first step in reconstructing a GB structure is to prepare the experimental HAADF-STEM images. This tutorial shows how to do this from scratch, following the steps described in 4.2:

1. Preparing the experimental HAADF-STEM images
   (a) **Geometric correction of image distortions**
   (b) **Scaling**
   (c) **Noise reduction by image averaging**
   (d) **Identify positions of atom columns**
   (e) **Initial image alignment along the rotation axis**

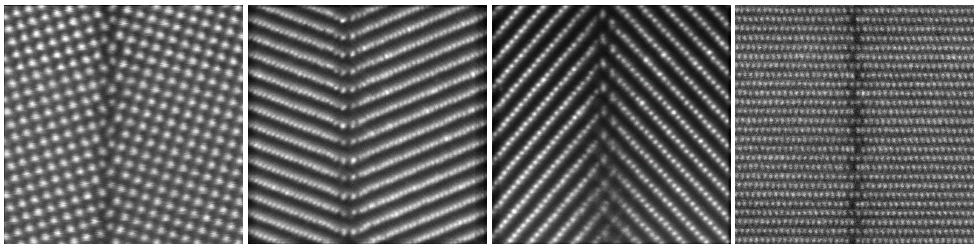
Step 1a: Geometric Correction of Image Distortions

The first step in the image preparation is to correct distortions form sample drift during the imaging process. This is done by applying an affine transformation to the image, estimated from matching the distorted atom column positions with a template generated from theoretical bulk parameters. For more details, see Section 4.2.1.

1. Identify atom column positions
2. Create ideal bulk template
3. Match template and atom columns
4. Estimate affine transformation

Original HAADF-STEM images from the YSZ bicrystal are included in the module `test_data` and are loaded into a list of NumPy 2-D arrays.

```
import grain_boundary_reconstruction.test_data as test_data
import grain_boundary_reconstruction.tools as tools

images_original_list =  test_data.get_original_images() #returns four
    example images in a list of numpy arrays
for i in range(0,4):
        image = images_original_list[i]
        tools.plot_image_array(image, axis=False) #plots 2-D array
```



(a) Image [001] **(b)** Image [1$\bar{3}$4] **(c)** Image [1$\bar{3}$2] **(d)** Image [1$\bar{3}$0]

Figure B.1: Original experimental HAADF-STEM images of the YSZ bicrystal from the module `test_data`.

1. Identifying Atom Column Positions

Throughout the image preparation part of this tutorial, the methods will be demonstrated by application to Image [001]. The positions of the atom columns in the experimental image is found by identifying all image intensity peaks above a specified limit, with a criterion of minimum distance.

```
import grain_boundary_reconstruction.geometric_correction as corr
import grain_boundary_reconstruction.peakFinder as peakFinder

image = images_original_list[0]

#parameters to be adjusted:
lower_intensity_limit_factor=0.8 #factor of max image intensity in percent
minimum_distance=12 #minimum peak distance in pixels

initial_peaks_df = peakFinder.find_maxima(image,
    lower_intensity_limit_factor, minimum_distance, plot=True)
```

The result is seen in Figure B.2. Not all columns are identified with this choice of parameters `lower_intensity_limit_factor` and `minimum_distance`. However, in this initial step, only (some of) the bulk atom column positions are needed. The coordinates

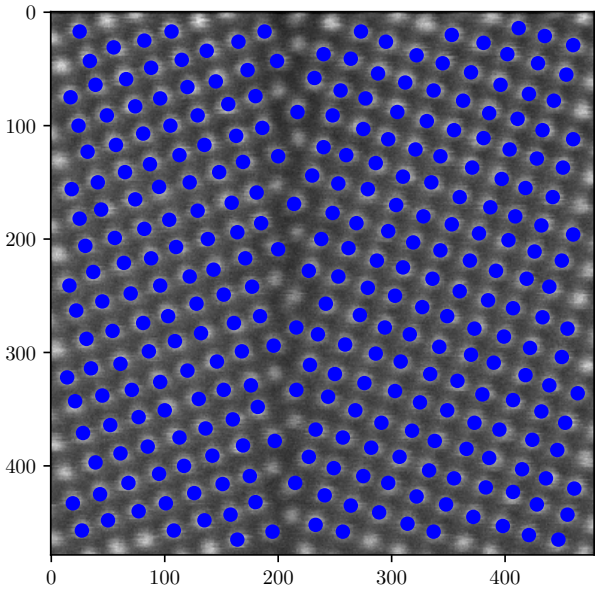


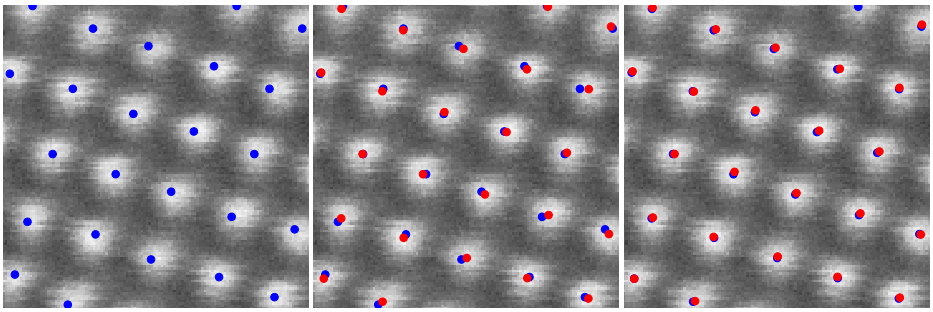
Figure B.2: Initial peaks

are stored in a *pandas* DataFrame in attributes z and x_proj referring to axis normal to and parallel with the GB plane, respectively.

```
initial_peaks_df
Out[1]:
        z  x_proj
0     115      42
1     105     100
..    ...     ...
289   216     278
290   196     294

[291 rows x 2 columns]
```

Next, the initial peak positions are refined by fitting a 2-D Gaussian function to an area if size $(2 \times atomradius)^2$. This is repeated until the peak positions appear to be stable, i.e. are not moving much from one iteration to the next.

```
atomradius=10
refined_peaks_df = peakFinder.fit_gaussians_to_maxima(image,
    initial_peaks_df, atomradius)
Out[2]:
Number of atoms detected: 291
```



(a) Initial peaks **(b)** Refined first time **(c)** Refined second time

Figure B.3: Peak refinement. In each figure, blue dots indicates the peaks found in the previous step, and red dots show refined positions. Second refinement gives little change in peak positions, meaning that a stable position is found.

2. Create Ideal Bulk Template

Next, we want to match these image points with points from a template with the wanted structure. The template is created by performing 2-D projections of the 3-D ideal structure seen in Figure B.4. A DataFrame with coordinates of an ideal FCC structure can be found in the `test_data` module. this structure is rotated and cut so that the GB plane {310} is normal to the z axis.

```
ideal_bulk_df = test_data.get_ideal_bulk()
tools.plot3Dstructure(ideal_bulk_df, markerSize=10, panes=True, figSize =
    (6,6), xlim=None, ylim =None, zlim=None, alpha=0.7)
```

2-D templates are then created from projections in directions corresponding to the experimental images, plotted in Figure B.5. Since the sample is a symmetric-tilt GB model, templates for the right and left side of the GB plane are identical copies except for mirrored z coordinates.

```
projection_angles_deg = [0, 38.3, 57.7, 90]
#creating list of template DataFrames for each side of the GB plane:
template_list_left = corr.create_bulk_templates(ideal_bulk_df,
    projection_angles_deg, side='left')
template_list_right = corr.create_bulk_templates(ideal_bulk_df,
    projection_angles_deg, side='right')
```

Match template and atom columns

The goal is now to match the template and the atom columns in the original image. Initially, we match one layer (edge layer) of atom columns. Left and right side is considered

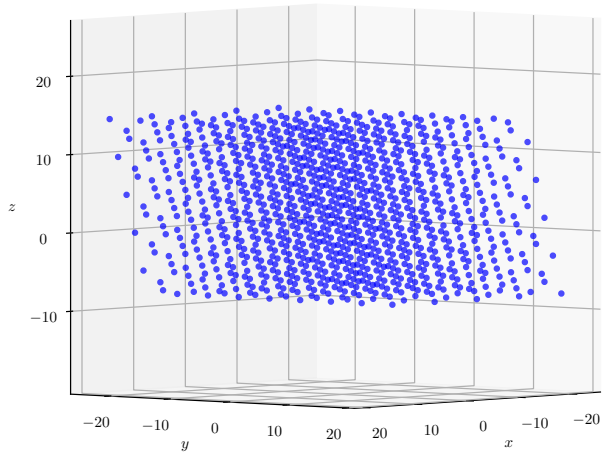


Figure B.4: Plot of ideal FCC structure from `test_data`.

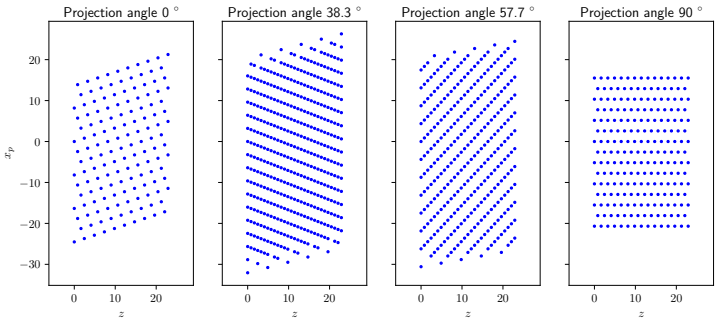


Figure B.5: Left-side templates created from 2-D projections of the ideal FCC structure in Figure B.4. Axes unit is Å.

separately. First, left side:

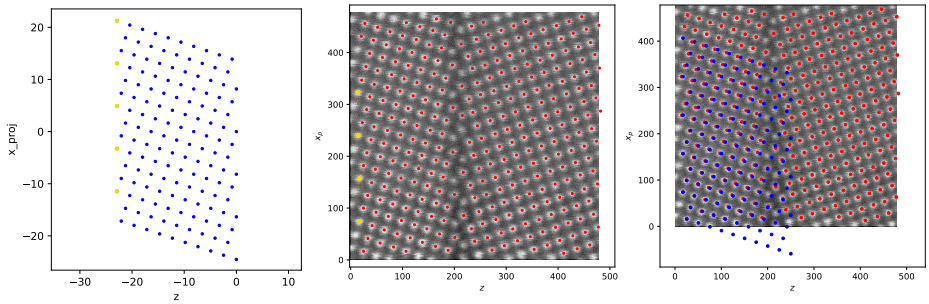
```
#load atom column positions found in the original images
refined_peaks_list=test_data.get_refined_peaks_original()
refined_peaks_df=refined_peaks_list[i]

side='left'

##extract layer from template
template_df=template_list_left[i]
```

```
a=5.17
dz_theor=a/(2*np.sqrt(10))
template_edge_df= corr.find_layer(template_df, side, dz=dz_theor)
tools.plot_2D_df(template_df, withAxis=True, figSize=(4,4), markerSize =
    5, color='b')
plt.scatter(template_edge_df.z, template_edge_df.x_proj, s=10, color='gold'
    )

##extract layer from image
z_layer_image_df= corr.find_layer(refined_peaks_df, side, dz=atomradius/2)

tools.plot_image_array(image)
plt.scatter(refined_peaks_df.z, refined_peaks_df.x_proj, s=6, color='r')
plt.scatter(z_layer_image_df.z, z_layer_image_df.x_proj, s=20, color='gold'
    )
```

We now find the scaling and translation that must be done to the template to match it to the initial points in the image. Note - here, we simultaneously get the scale of the image! The variable `scale_factor_x_proj` printed below is the pixel-to-Ångstrom ratio of the image. This is calculated from the few atom columns chosen in at this step. Remember, when we later perform the affine transformation on the image, the whole image is scaled to fit this template. Hence, this is the correct image scale, *after* the affine transformation is applied.

```
scale_factor_x_proj, translation_x_proj, translation_z  = corr.
    match_initial_points(template_edge_df, z_layer_image_df, match_atoms='
    all')

scale_factor_x_proj
Out[3]:
10.17110651592613

#Scaling and translating template:
template_scaled_df = template_df.copy()
template_scaled_df*=scale_factor
template_scaled_df.x_proj+=translation_x_proj
template_scaled_df.z+=translation_z

tools.plot_image_array(image)
plt.scatter(refined_peaks_df.z, refined_peaks_df.x_proj, color='r' , s=10)
plt.scatter(template_scaled_df.z, template_scaled_df.x_proj, color='b', s
    =10 )
```

Figure B.6 shows the result of the initial matching. Now the template must be cut so that no atoms on the wrong side of the GB is included in the affine transformation astimation. Then, the atom columns in the template are paired - in the affine transformation estimation, we must know which atom columns we want to match. The function `match_two_point_sets()` returns two DataFrames that the index of the atom points



(a) Template **(b)** Image **(c)** Iniital matching

Figure B.6: Matching template to image by finding the scale and translation needed to move the yellow dots marked in **(a)** to corresponding red dots in **(b)**. The result is shown in **(c)**. This plot makes the deviation from an ideal structure obvious. Axis in **(a)** have the unit Ångstrom while **(b)** and **(c)** has pixels.

match. That is, first row in `template_matched_df` and first row in `image_matched_df` are one "pair".

```
#Cut template
max_z_value=190
template_scaled_df=template_scaled_df.ix[template_scaled_df.z<max_z_value].
    copy()

template_matched_df,image_matched_df = corr.match_two_point_sets(
    template_scaled_df, refined_peaks_df, plot=True)
```

The Affine matrix coefficients can now be estimated, using `transform.estimate_transform()` in `skimage`[102] - see Appendix A for details on this implementation. From the `geometric_correction` module:

```
def estimate_affine_matrix(template_matched_df,image_matched_df):
    '''estimate affine transformation matrix. Returns matrix as numpy array
        '''
    src = template_matched_df.as_matrix(['z', 'x_proj'])
    dst = image_matched_df.as_matrix(['z', 'x_proj'])
    tform = transform.estimate_transform('affine', src, dst)
    affine_matrix=np.array(tform.params)
    return affine_matrix
```

In the main file:

```
affine_matrix_left = corr.estimate_affine_matrix(template_matched_df,
    image_matched_df)

affine_matrix_oneside
Out[4]:
array([[ 0.98118532, -0.01226085,  0.67416634],
       [ 0.01673407,  0.99936376, -0.38301461],
       [ 0.        ,  0.        ,  1.        ]])
```

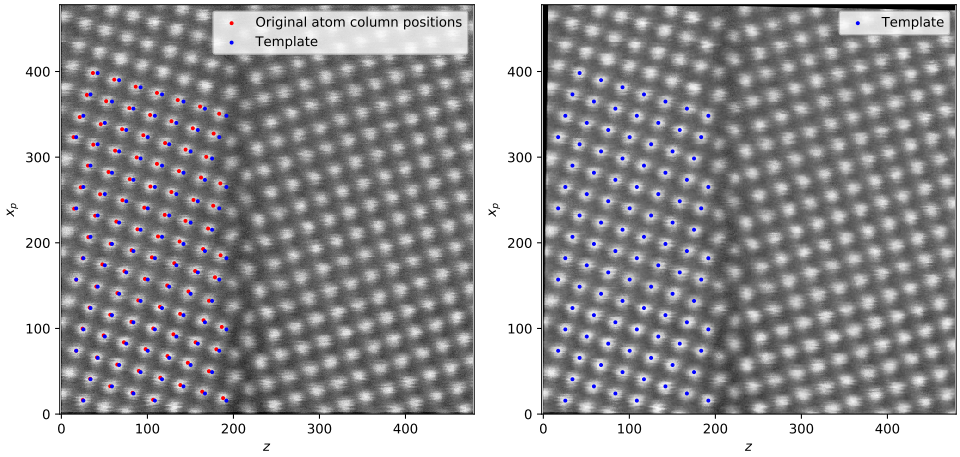
The effect of the transformation can be calculated from of the matrix coefficients, see Equations (A.6):

```
corr.get_coeffinients(affine_matrix_left)
Out[5]:
Rotation angle T: 0.977082 degrees

Shear angle C (x-shear): -0.274175 degrees

Scale factors [sx, sy]:
 [0.981328, 0.999439]

Translation vector [tx, ty]:
 [0.674166, -0.383015]

Returns [sx, sy, T, C, tx, ty]
```

```
##APPLY
image_transformed = corr.apply_affine_transform(affine_matrix_left, image)

#plot image and transformed image with templates
tools.plot_image_array(image)
plt.scatter(image_matched_df.z, image_matched_df.x_proj, s=4, color='r',
    label="Original atom column positions")
plt.scatter(template_matched_df.z, template_matched_df.x_proj, s=4, color='
    b', label="Template")
plt.legend()
tools.plot_image_array(image_transformed)
plt.scatter(template_matched_df.z, template_matched_df.x_proj, s=4, color='
    b', label="Template")
plt.legend()
```

Figure B.7 shows `affine_matrix_left` applied to the image. As we see from the rotation angle calculated from the coefficients, a 0.977° rotation was needed to match the template, which has a vertical {310} plane. In the production of the bicrystal sample, the aim for the misorientation angle between the two single crystals was $2\theta = 18.43°$, as this will give the {310} GB plane. Some small deviation in the angle might howevever be present. Therefore, the whole process should be repeated for the right side of the bulk, giving a second affine matrix `affine_matrix_right`. This is not shown here. The average of the two matrices is the transformation that is finally applied to the image:

```
#Calculate average affine transform
affine_matrix_avg=(affine_matrix_right+ affine_matrix_left)/2
##APPLY
image_transformed = corr.apply_affine_transform(affine_matrix_avg, image)
```



(a) Before affine transformation **(b)** After affine transformation

Figure B.7: Image **(a)** before and **(b)** after affine transformaiton, plotted along with original atom positions (red) and template (blue) to see the improvement.

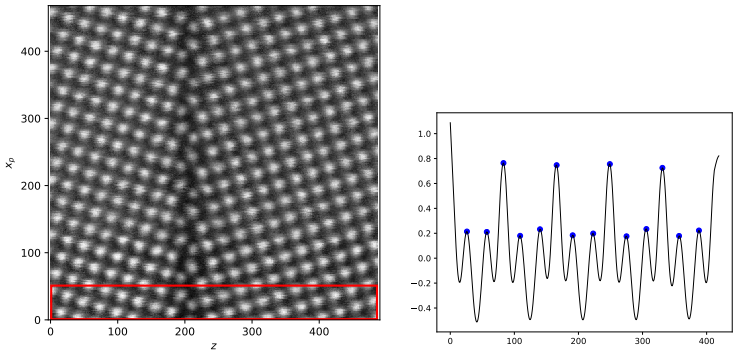
Step 1c: Noise reduction by image averaging

For noise reduction, the image is averaged over several GB units, which are located using cross-correlation between the image and a smaller section of the same image, called a template, see Figure B.8a.

```
import grain_boundary_reconstruction.noise_reduction as nr

#get list of images as 2-D arrays (from .png)
images_affined_list =  nr.get_affined_images()
i=0
image = images_affined_list[i]

##get indices (shifts in pixels) for cross-correlation maxima
max_ind = nr.cross_correlation_along_GB(image, plot=True, template_height
    =50)
```

Figure B.8b shows the calculated cross-correlation function (CCF). The maxima corresponds to high similarity between image and template. The four highest peaks are chosen and the image is averaged over these locations, resulting in the image shown in Figure B.9.

```
##choose which peaks/shifts to make averaged image from
peaks=np.array([2,5,8,11])
shifts= max_ind[peaks]
averaged_image_height=150
averaged_image, no_units = nr.create_averaged_image(image, shifts,
    averaged_image_height)
```



(a) Image and template (red rectangle) **(b)** CCF

Figure B.8: Cross-correlation between image and template. The cross-correlation function (CCF) in **(b)** is calculated from Equation (4.5). The horizontal axis has units in pixels, indicating the shift of the template across the image along the vertical axis.

```
tools.plot_image_array(averaged_image)
```

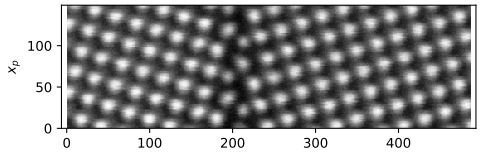


Figure B.9: Image [001] averaged over 4 GB units.

Step 1d: Identify positions of atom columns

This step is similar to some of the steps in Section B.2. Here, some code and output is repeated to show the improvement of peak stability after image averaging. Now, the parameters I_l (`lower_intensity_limit_factor`) and d_{min} (`minimum_distance`) must be adjusted so that all peaks are identified.

```
images_averaged_list =  test_data.get_averaged_images() #returns list of
    images as 2-D arrays (from .tif)
i=0
image = images_original_list[i]

#parameters to be adjusted:
lower_intensity_limit_factor=0.5 #factor of max image intensity in percent
minimum_distance=12 #minimum peak distance in pixels
initial_peaks_df = peakFinder.find_maxima(image,
    lower_intensity_limit_factor, minimum_distance)
```

```
atomradius=12
#refinement of initial peaks
refined_peaks_df = peakFinder.fit_gaussians_to_maxima(image,
    initial_peaks_df, atomradius, iterations=4)

Out[5]:
Number of atoms detected: 118
Average position move: 1.08901286832
Number of atoms detected: 118
Average position move: 0.710460552915
Number of atoms detected: 117
Average position move: 0.36802426967
Number of atoms detected: 117
Average position move: 0.418488885613
```

The peak-refining function prints "Average position move", which tells how many pixels each peak position is changed on average. This is seen to stabilize at around 0.4 pixels. One atom close to the edge disappears, which is why "Number of atoms detected" goes from 118 to 117.

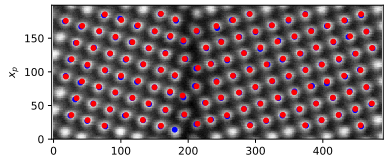


Figure B.10: Image [001] with initial peaks (blue) and refined peaks(red)

The atom column positions are plotted on top of the image in Figure B.10. These coordinates, along with the coordinates for atom column positions in Image [1$\bar{3}$4], Image [1$\bar{3}$2] and Image [1$\bar{3}$0], is the starting point for the 3-D reconstruction.

Appendix C

Code

C.0.0.1 tools.py

```
"""
### tools.py ###

Tools for plotting, reading and writing files, kernel density estimation,
    NN distances, matching point sets for RMSD estimation etc.
"""
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from mpl_toolkits.mplot3d import proj3d
from sklearn.cluster import DBSCAN
import time
from scipy import ndimage
from sklearn.neighbors import KDTree
from sklearn.neighbors import KernelDensity
import random

#####

def plot_image_array(image, axis=True, filter_factor=0, title=False,
    figHeight=5, cmap='Greys_r'):
    '''Plot image array'''
    x=np.arange(0, np.shape(image)[1])
    y=np.arange(0, np.shape(image)[0])
    X,Y=np.meshgrid(x,y)
    imratio=image.shape[0]/image.shape[1]
    fig,ax = plt.subplots(1, figsize=(figHeight, imratio*figHeight))
    image_filtered= ndimage.filters.gaussian_filter(np.asarray(image),
        filter_factor)
    ax.pcolormesh(X,Y,image_filtered, cmap=cmap)
    plt.axis('equal')
    plt.subplots_adjust(left=0.12,  right=0.99, bottom=0.12, top=0.99,
        wspace=0, hspace=0)
```

```
    plt.xlabel('$z$')
    plt.ylabel('$x_p$')

    if axis is not True:
        plt.axis('off')
        plt.subplots_adjust(left=0,  right=1, bottom=0, top=1, wspace=0,
            hspace=0)

    if title is not False:
        plt.subplots_adjust(left=0.12,  right=0.99, bottom=0.12, top=0.9,
            wspace=0, hspace=0)
        plt.title(title)
    return fig, ax

def plot_2D_df(projection_2D_df, withAxis=True, figSize=(4,4), markerSize
    = 10, color='b'):
    '''Plot 2D reprojection'''
    fig, ax = plt.subplots(figsize=figSize)
    ax.scatter(projection_2D_df.z, projection_2D_df.x_proj,  s=markerSize,
        color=color)
    ax.axis('equal')
    #ax.set_ylim(ax.get_ylim()[::-1])

    if withAxis:
        ax.set_xlabel('z')
        ax.set_ylabel('x_proj')
        fig.subplots_adjust(left=0.16,  right=0.98, bottom=0.11, top=0.98,
            wspace=0, hspace=0)
    else:
        ax.axis('off')
        fig.subplots_adjust(left=0,  right=1, bottom=0, top=1, wspace=0,
            hspace=0)

def plot3Dstructure(atompos_df, df_2=None, markerSize=10, view_dir=0,
    orthogonal=True,linewidths=0.5,elev=0, panes=False, figSize = (6,6),
    xlim=None, ylim =None, zlim=None, alpha=0.7, colors=None, c=[0,0,1], c2
    =[1,0,0]):
    xlim = xlim or [np.min(atompos_df.x),np.max(atompos_df.x)]
    ylim = ylim or [np.min(atompos_df.y),np.max(atompos_df.y)]
    zlim = zlim or [np.min(atompos_df.z),np.max(atompos_df.z)] #bestemmer
        utifraa df size OR arguments

    def axisEqual3D(ax):
        extents = np.array([getattr(ax, 'get_{}lim'.format(dim))() for dim
            in 'xyz'])
        sz = extents[:,1] - extents[:,0]
        centers = np.mean(extents, axis=1)
        maxsize = max(abs(sz))
        r = maxsize/2
        for ctr, dim in zip(centers, 'xyz'):
            getattr(ax, 'set_{}lim'.format(dim))(ctr - r, ctr + r)

    if orthogonal:
        def orthogonal_proj(zfront, zback):
            a = (zfront+zback)/(zfront-zback)
            b = -2*(zfront*zback)/(zfront-zback)
            return np.array([[1,0,0,0],
```

```
                                [0,1,0,0],
                                [0,0,a,b],
                                [0,0,0,zback]])

        proj3d.persp_transformation = orthogonal_proj

    if  colors is not None:
        if colors=='pair':
            colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1,
                len(atompos_df.z))]
            random.shuffle(colors)
        rgba_colors=colors
        rgba_colors_2=colors

    else:
        rgba_colors = np.zeros((atompos_df.shape[0],4))
        rgba_colors[:,0] = c[0] #red
        rgba_colors[:,1] = c[1] #green
        rgba_colors[:,2] = c[2] #blue
        alphas =np.ones(atompos_df.shape[0])*alpha
        rgba_colors[:, 3] = alphas

        if df_2 is not None:
            if colors is None:
                rgba_colors_2 = np.zeros((df_2.shape[0],4))
                rgba_colors_2[:,0] = c2[0] #red
                rgba_colors_2[:,1] = c2[1] #green
                rgba_colors_2[:,2] = c2[2] #blue
                alphas =np.ones(df_2.shape[0])*alpha
                rgba_colors_2[:, 3] = alphas

    fig = plt.figure(figsize=figSize)
    ax = fig.add_subplot(111, projection='3d')
    ax.view_init(elev=elev, azim=270+view_dir)
    ax.scatter(atompos_df.x,
               atompos_df.y,
               atompos_df.z,
               s=markerSize, marker='o',c=rgba_colors, edgecolors='k',
                   linewidths=linewidths, zdir='z')

    if df_2 is not None:
        ax.scatter(df_2.x,
                   df_2.y,
                   df_2.z,
                   s=markerSize, marker='o',c=rgba_colors_2, edgecolors='r'
                       ,linewidths=linewidths, zdir='z')

    #####Layout####
    ax.set_xlim3d(xlim)
    ax.set_ylim3d(ylim)
    ax.set_zlim3d(zlim)
    axisEqual3D(ax)
    plt.subplots_adjust(left=0,  right=1, bottom=0, top=1, wspace=0, hspace
        =0)

    if panes==False:
        #Get rid of panes
```

```
        ax.w_xaxis.set_pane_color((1.0, 1.0, 1.0, 0.0))
        ax.w_yaxis.set_pane_color((1.0, 1.0, 1.0, 0.0))
        ax.w_zaxis.set_pane_color((1.0, 1.0, 1.0, 0.0))
        # Get rid of the spines
        ax.w_xaxis.line.set_color((1.0, 1.0, 1.0, 0.0))
        ax.w_yaxis.line.set_color((1.0, 1.0, 1.0, 0.0))
        ax.w_zaxis.line.set_color((1.0, 1.0, 1.0, 0.0))
        # Get rid of the ticks
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_zticks([])
    else:
        ax.set_xlabel('$x$')
        ax.set_ylabel('$y$')
        ax.set_zlabel('$z$')
    return fig, ax

def find_projected_coordinates(atompos_df, proj_angle, side_tilt=0):
    ''' Takes df with list of 3D coordinates (x_in, y_in, z_in) where z
        axis is normal to GB plane
    Returns df with 2D coordinates (x_proj, z_in), a projection in wanted
        direction proj_angle (input in degrees)'''

    #projecting matrix
    theta = np.deg2rad(proj_angle)
    #Projection matrix
    P = np.array([[np.cos(theta), np.sin(theta), 0], [0, 0, 1]])
    # using equation  x_proj = x \cos \theta + y \sin \theta. z=z.

    #tilt about x_proj axis - matrix
    cos=np.cos(np.deg2rad(side_tilt))
    sin=np.sin(np.deg2rad(side_tilt))
    ux=np.cos(theta)
    uy=np.sin(theta)
    uz=0

    R_x_proj = np.array([[cos+ux*ux*(1-cos),ux*uy*(1-cos)-uz*sin, ux*uz*(1-
        cos)+uy*sin],
                [ux*uy*(1-cos)-uz*sin,cos+uy*uy*(1-cos), uy*uz*(1-cos)+
                    ux*sin],
                [ux*uz*(1-cos)-uy*sin, ux*uz*(1-cos)+ux*sin, cos+uz*uz
                    *(1-cos)]])

    xyz = np.array(atompos_df.as_matrix(['x', 'y', 'z'])).T

    #rotate
    x,y,z=R_x_proj  xyz
    #project
    x_proj, z = P  np.array([x,y,z])
    projection_df = pd.DataFrame({'x_proj': x_proj, 'z': z})
    return projection_df

def get_column(structure_df, coord_z, coord_x_proj, angle_deg, radius=0.2):
    '''returns df with atoms in the column specified in coords=[x_proj,z]
        in projectionimage'''
    a=5.17
    dz=a/(2*np.sqrt(10))
```

```
    #get right layer
    column_df = structure_df[structure_df.z<coord_z+dz/2].copy()
    column_df =column_df.drop(column_df.index[column_df.z<coord_z-dz/2])

    #test alle atom i this layer, sjekk om dei er naert nok using equation
         x_proj = x \cos \theta + y \sin \theta
    for ind in column_df.index:
        x=column_df.x[ind]
        y=column_df.y[ind]
        x_proj = x*np.cos(np.deg2rad(angle_deg)) + y*np.sin(np.deg2rad(
            angle_deg))

        if abs(x_proj-coord_x_proj)>radius:
            column_df=column_df.drop(ind)
    return column_df

def cluster_2D_projection(atompos_proj_df, max_dist): #atompos_proj_df has
    x_proj, z-coordinates
    '''Performs clustering on 2D projected df,
    returns center_df with sixes ad labe'''

    time_start=time.time()
    z = np.array(atompos_proj_df.z)
    x_proj = np.array(atompos_proj_df.x_proj)

    X2d = np.array([ [z[i], x_proj[i]] for i in range(0, len(z)) ])
    db = DBSCAN(eps=max_dist, min_samples=1).fit(X2d)

    labels=db.labels_
    # Number of clusters in labels, ignoring noise if present.
    n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

    ##Reknar ut center som CM av kvar cluster..
    center_list = []
    for i in range(0, n_clusters_):
        atom=X2d[labels==i]
        sizes = np.sum(labels==i)
        x_proj_mean = np.mean(atom[:,1])
        z_mean = np.mean(atom[:,0])
        dict1 = {'x_proj': x_proj_mean, 'z': z_mean, 'sizes': sizes, 'labe'
            : i }
        center_list.append(dict1)

    center_db_df = pd.DataFrame(center_list, columns=['x_proj', 'z', 'sizes
        ', 'labe'])
    print("Found %d clusters \n Clustering on %d points \n Time: %f minutes
         \n "  %(n_clusters_, X2d.shape[0],(time.time()-time_start)/60  ))
    return center_db_df

def label_2D_projection(atompos_proj_df, max_dist):
    '''Performs clustering on 2D projected df,
    returns copy of df in with labels labe and sizes of cluster'''

    z = np.array(atompos_proj_df.z)
    x_proj = np.array(atompos_proj_df.x_proj)

    X2d = np.array([ [z[i], x_proj[i]] for i in range(0, len(z)) ])
```

```
    db = DBSCAN(eps=max_dist, min_samples=1).fit(X2d)

    labels=db.labels_

    sizes_list = []
    for i in range(0, len(labels)):
        sizes = np.sum(labels==labels[i])
        sizes_list.append(sizes)

    atompos_proj_df_with_labels=atompos_proj_df.copy()
    atompos_proj_df_with_labels['labe']=labels
    atompos_proj_df_with_labels['sizes']=sizes_list
    return atompos_proj_df_with_labels

def cluster_3D(overlap_df, minsample=1, max_dist = 2):
    '''Returns DataFrame with clusters found by DBSCAN'''

    X3d = np.array([np.array(overlap_df.z), np.array(overlap_df.y), np.
        array(overlap_df.x)]).T
    db = DBSCAN(eps=max_dist, min_samples=minsample).fit(X3d)  ###kan evt
        justere min_samples til mykje hoegare! max size rundt 80 (90) paa
        atoma

    labels=db.labels_
    # Number of clusters in labels, ignoring noise if present.
    n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

    ##Reknar ut center som CM av kvar cluster
    center_list = []
    for i in range(0, n_clusters_):
        atom=X3d[labels==i]
        sizes = np.sum(labels==i)
        z_mean = np.mean(atom[:,0])
        y_mean = np.mean(atom[:,1])
        x_mean = np.mean(atom[:,2])
        dict1 = {'atom': 'Zr', 'x': x_mean, 'y': y_mean, 'z': z_mean, '
            sizes': sizes }
        center_list.append(dict1)

    center_db_df = pd.DataFrame(center_list, columns=['atom','x', 'y', 'z',
         'sizes'])
    print("Found %d clusters \n Clustering on %d points \n Minsample= %d \n
         "  %(n_clusters_, overlap_df.z.size, minsample  ))
    return center_db_df

def printXYZ(center_db_df, filename, sizes=True, header=None): #tek inn df
    med x, y, z, size. filename uten extention
    '''Takes filename without extention. creates df with x,y,z,sizes
        entries, if sizes=true... . '''
    #heller lage func for aa lage xyz om til crystalmakervennlege filer (
        utan sizes) seinare
    file = open(filename+'.xyz','w')

    no_atoms=len(center_db_df.z)
    file.write(str(no_atoms) + "\n")  #numbers of atoms

    #to avoid index mistakes
```

```
        xs=np.array(center_db_df.x)
        ys=np.array(center_db_df.y)
        zs=np.array(center_db_df.z)
        atoms=np.array(center_db_df.atom)

        if sizes:
            header = header if header is not None else "Reconstruction, xyz-
                file with size"
            file.write(header+"\n")
            col_names_line = "atom x y z sizes"
            file.write(col_names_line+"\n")
            sizes=np.array(center_db_df.sizes)

            for i in range(no_atoms):
                pos_line = atoms[i]+" %f %f %f %d\n"  %(xs[i], ys[i], zs[i],
                    sizes[i])
                file.write(pos_line)
        else:
            header = header if header is not None else  "Reconstruction, xyz-
                file without size"
            file.write(header+"\n")
            col_names_line = "atom x y z"
            file.write(col_names_line+"\n")
            for i in range(no_atoms):
                pos_line = atoms[i]+" %f %f %f\n"  %(xs[i], ys[i], zs[i])
                file.write(pos_line)

        file.close()

    def readXYZ(name, path=None, print_col_names=True):
        '''Takes filename without extentions, with atomname x y z and maybe
            size.'''

        filename = path+'/'+name if path is not None else name

        f = open(filename+'.xyz', 'r')
        for i in range(3):
            line = f.readline()
        f.close()

        if len(line.split(' ')) == 5:
             atompos_df = pd.read_csv(filename+'.xyz', sep=" ", header=2,
                 index_col=False, names = ['atom', 'x', 'y', 'z', 'sizes'])
        elif len(line.split(' ')) == 4:
            atompos_df = pd.read_csv(filename+'.xyz', sep=" ", header=2,
                index_col=False, names = ['atom', 'x', 'y', 'z'])
        else:
            print('Check file format')
            atompos_df = 0
        if print_col_names:
            print(atompos_df.columns)
        return atompos_df

    def printCIF(filename, ideal_df, cell_length_a, cell_length_b,
        cell_length_c ):
        '''Takes xyz, transform to cif files with fraction coordinates.'''
        x_min = np.min(ideal_df.x)
```

```
    y_min = np.min(ideal_df.y)
    z_min = np.min(ideal_df.z)
    angle=90.0

    file = open(filename+'.cif','w')
    file.write("data_cif\n")
    file.write("_cell_length_a\t%f\n"%(cell_length_a))
    file.write("_cell_length_b\t%f\n"%(cell_length_b))
    file.write("_cell_length_c\t%f\n"%(cell_length_c))
    file.write("_cell_angle_alpha\t%f\n"%(angle))
    file.write("_cell_angle_beta\t%f\n"%(angle))
    file.write("_cell_angle_gamma\t%f\n"%(angle))
    file.write("loop_\n")
    file.write("_atom_site_label\n")
    file.write("_atom_site_fract_x\n")
    file.write("_atom_site_fract_y\n")
    file.write("_atom_site_fract_z\n")

    atoms=np.array(ideal_df.atom)
    xs=np.array(ideal_df.x)
    ys=np.array(ideal_df.y)
    zs=np.array(ideal_df.z)

    for i in range(len(ideal_df.index)):
        file.write(atoms[i]+"\t%f\t%f\t%f\n"%((xs[i]-x_min)/cell_length_a,
                                              (ys[i]-y_min)/cell_length_b,
                                              (zs[i]-z_min)/(cell_length_c)))
    file.close()
    print("File created:\n"+filename)

def clusterCIF(atompos_df, cell_length_a, cell_length_b, cell_length_c,
    minsample=1, max_dist=0.0001):
    '''takes big df, creates unit with specified cell lengths. This gives
        many atoms in the same spot. Does clustering before printing to cif
        -file.'''

    real_rec_fract_df = atompos_df.copy()
    x_min=np.min(real_rec_fract_df.x)
    y_min=np.min(real_rec_fract_df.y)
    z_min=np.min(real_rec_fract_df.z)

    real_rec_fract_df.x = (real_rec_fract_df.x-x_min)/(cell_length_a)
    real_rec_fract_df.y = (real_rec_fract_df.y-y_min)/(cell_length_b)
    real_rec_fract_df.z = (real_rec_fract_df.z-z_min)/(cell_length_c) #just
         to avoid weird edges
    bit_df=real_rec_fract_df.copy()

    n_x_units=int(np.max(real_rec_fract_df.x)+1)
    n_y_units=int(np.max(real_rec_fract_df.y)+1)

    for i in range(1, n_x_units):
        unit_inds = [np.array(np.array([real_rec_fract_df.x<(i+1)])*np.
            array([real_rec_fract_df.x>=i]))][0][0]
        bit_df.x[unit_inds] = real_rec_fract_df.x[unit_inds]-i

    for i in range(1, n_y_units):
```

```
            unit_inds = [np.array(np.array([real_rec_fract_df.y<(i+1)])*np.
                array([real_rec_fract_df.y>=i]))][0][0]
            bit_df.y[unit_inds] = real_rec_fract_df.y[unit_inds]-i

        real_rec_clustered_df = cluster_3D(bit_df, minsample=minsample,
            max_dist=max_dist)

        real_rec_clustered_df.x=(real_rec_clustered_df.x*(cell_length_a)+x_min)
        real_rec_clustered_df.y=(real_rec_clustered_df.y*(cell_length_b)+y_min)
        real_rec_clustered_df.z=(real_rec_clustered_df.z*(cell_length_c)+z_min)
        return real_rec_clustered_df

    def readCIF(filename, n_units=1):
        '''takes big df, creates unit with specified cell lengths. This gives
            many atoms in the same spot. Does clustering before printing to cif
            -file.'''

        f = open(filename+'.cif', 'r')
        line = f.readline()
        line = f.readline()
        cell_length_x=float(line.split('\t')[1][:-1])
        line = f.readline()
        cell_length_y=float(line.split('\t')[1][:-1])
        line = f.readline()
        cell_length_z=float(line.split('\t')[1][:-1])
        f.close()

        atompos_df = pd.read_csv(filename+'.cif', sep="\t", header=11,
            index_col=False, names = ['atom', 'x', 'y', 'z'])
        atompos_df.x*=cell_length_x
        atompos_df.y*=cell_length_y
        atompos_df.z*=cell_length_z

        #multiply
        unit_df_big=atompos_df.copy()
        for i in range(0, n_units):
            for j in range(0, n_units):
                shifted_unit_df=atompos_df.copy()
                shifted_unit_df.x+=i*cell_length_x
                shifted_unit_df.y+=j*cell_length_y

                unit_df_big=pd.concat([unit_df_big, shifted_unit_df])

        #remove overlapping atoms
        minsample=1 #cluster: find mean if three intersections are close.
        unit_df_big_clustered = cluster_3D(unit_df_big, minsample, max_dist
            =0.01)
        return unit_df_big_clustered

    def rotate_3D(atompos_df, rotation_angle_deg=90, center_x=None, center_z=
        None):
        '''Tilting crystal. Takes angle (in deg) atompos.df with columns x, y,
            z, and rotates about center of df (rotation axis y)
        z_r = z_0 \cos \theta_r - x_0 \sin \theta_r
        x_r = x_0 \cos \theta_r + z_0 \sin \theta_r
        $z_0, x_0$ is input coordinates and $\theta_r$ is the rotation angle
            about the center of the image.
```

```
    '''
    atompo_df=atompos_df.copy()

    #shift origin to center of image
    center_x = center_x if center_x is not None else np.mean(atompo_df.x)
    center_z = center_z if center_z is not None else np.mean(atompo_df.z)

    atompo_df.x=atompo_df.x-center_x
    atompo_df.z=atompo_df.z-center_z

    theta_rotate = np.deg2rad(rotation_angle_deg)

    A = np.array([[np.cos(theta_rotate), -np.sin(theta_rotate)],
         [np.sin(theta_rotate), np.cos(theta_rotate)]])

    #calculated rotated point coordinates
    w = np.array([atompo_df.z, atompo_df.x])
    w_rotated = np.linalg.solve(A,w)
    #shift origin back
    atompos_rotated_df = pd.DataFrame({ 'x': w_rotated[1]+center_x, 'y':np.
        array(atompo_df.y), 'z': w_rotated[0]+center_z, })
    return atompos_rotated_df

def match_two_point_sets_3D(template_df, image_df, plot=False):
    '''Point pair: one in template_df and one in image_df CLOSER than
        smallest_dist/2 (smallest dist within template)
    returns dfs matched by index. Template input must be scaled and matched
        . '''

    template_array = template_df.as_matrix(columns = ['x', 'y', 'z'])
    kdt = KDTree(template_array,  metric='euclidean')
    dist, inds = kdt.query([[template_array[0][0],template_array[0][1],
        template_array[0][2]]], k=3)
    #smallest distance within template: Use half this as limit for NN
    smallest_dist = dist[0][1]
    image_array = image_df.as_matrix(columns = ['x','y', 'z'])
    kdt = KDTree(image_array,  metric='euclidean')

    #iterate over all atoms in template. find nearest neighbour in the
        image peaks. IF clore enough, add to list.
    template_points_with_neighbour=[]
    image_points_with_neighbour=[]
    for atom_ind in template_df.index:
            dist, inds = kdt.query([[template_df.x[atom_ind], template_df.y
                [atom_ind],template_df.z[atom_ind]]], k=1)

            if dist[0][0]<smallest_dist/2:
                dict_template = {'x':template_df.x[atom_ind], 'y':
                    template_df.y[atom_ind] ,'z': template_df.z[atom_ind] }
                dict_image = { 'x':image_array[inds[0][0]][0],  'y':
                    image_array[inds[0][0]][1], 'z': image_array[inds
                    [0][0]][2]}
                template_points_with_neighbour.append(dict_template)
                image_points_with_neighbour.append(dict_image)

    template_matched_df = pd.DataFrame(template_points_with_neighbour,
        columns=['x', 'y', 'z'])
```

```
    image_matched_df = pd.DataFrame(image_points_with_neighbour, columns=['
        x', 'y', 'z'])
    colors = [plt.cm.Spectral(each)
              for each in np.linspace(0, 1, len(image_matched_df.z))]
    random.shuffle(colors

    if plot:
        fig, ax = plot3Dstructure(image_matched_df, df_2=
            template_matched_df, colors=colors)
    return template_matched_df, image_matched_df

def rms_3D(df1, df2):
    '''calculates the RMSE between two sets of 3D data points'''
    return np.sqrt( np.sum((df1.z-df2.z)**2 +  (df1.x-df2.x)**2 +(df1.y-df2
        .y)**2  )/len(df1.z))

def kde1D(x, bandwidth=0.01):
    '''plots gaussian kernel density of 1D array'''
    X=np.array(x)
    X = X.reshape([-1,1])
    size_x=np.max(x)-np.min(x)
    X_plot = np.linspace(np.min(x)-size_x*0.05, np.max(x)+size_x*0.05,
        1000)[:, None]
    # Gaussian KDE
    kde = KernelDensity(kernel='gaussian', bandwidth=bandwidth).fit(X)

    # score_samples() returns the log-likelihood of the samples
    log_dens = kde.score_samples(X_plot)
    z = np.exp(log_dens)

    fig, ax = plt.subplots(1)
    ax.fill(X_plot[:, 0], z, fc='#AAAAFF')
    return fig,ax

def kde2D(x, y, bandwidth, xbins=100j, ybins=100j, **kwargs):
    """Build 2D kernel density estimate (KDE)."""

    # create grid of sample locations (default: 100x100)
    xx, yy = np.mgrid[x.min():x.max():xbins,
                      y.min():y.max():ybins]

    xy_sample = np.vstack([yy.ravel(), xx.ravel()]).T
    xy_train  = np.vstack([y, x]).T
    # Gaussian KDE
    kde = KernelDensity(kernel='gaussian', bandwidth=bandwidth, **kwargs)
    kde.fit(xy_train)

    # score_samples() returns the log-likelihood of the samples
    z = np.exp(kde.score_samples(xy_sample))
    return xx, yy, np.reshape(z, xx.shape)

def kde3D(x, y, z, bandwidth=0.01, xbins=100j, ybins=100j, zbins=100j, **
    kwargs):
    """Build 2D kernel density estimate (KDE)."""

    # create grid of sample locations (default: 100x100)
    xx, yy, zz = np.mgrid[x.min():x.max():xbins,
```

```
                    y.min():y.max():ybins,
                    z.min():z.max():zbins]

    xyz_sample = np.vstack([zz.ravel(), yy.ravel(), xx.ravel()]).T
    xyz_train  = np.vstack([z, y, x]).T
    # Gaussian KDE
    kde_skl = KernelDensity(kernel='gaussian', bandwidth=bandwidth, **
        kwargs)
    kde_skl.fit(xyz_train)

    # score_samples() returns the log-likelihood of the samples
    z = np.exp(kde_skl.score_samples(xyz_sample))
    return xx, yy, zz, np.reshape(z, xx.shape)

def investigate_neighbours_2D(atompos_df, plot_kde=True, max_dist=2,
    nhood_size=10, MIN_dist=0, n_neighbours=None):
    ''' Returns array of nearest neighbours distances. '''

    atompos_array = atompos_df.as_matrix(columns = ['x_proj', 'z'])
    kdt = KDTree(atompos_array,  metric='euclidean')
    n_neighbours=n_neighbours if n_neighbours is not None else len(
        atompos_df.z)

    dist_arr=np.array([])
    for i in range(0, len(atompos_df.z)):
        dist, ind = kdt.query([atompos_array[i]], k=n_neighbours)
        dist=dist[0]
        dist=dist[1:]
        dist=dist[dist<nhood_size]
        dist=dist[dist>MIN_dist]
        dist_arr = np.concatenate((dist_arr, dist))

    if plot_kde:
        fig,ax=kde1D(dist_arr, bandwidth=0.01)
    return layer_db_df, dist_arr

def investigate_neighbours_3D(atompos_df, check_sub_df=None, plot_hist=
    False, plot_kde=True,dbin = 1, max_dist=2, MIN_dist=0, nhood_size=10,
    n_neighbours=5):
    ''' Plots histogram with all neighbour distances, and layer means
    Returns layer_db_df with mean and std of each "layer" of neighbours
    if check_sub_df is input, only these atoms are checked. '''

    atompos_array = atompos_df.as_matrix(columns = ['x', 'y', 'z'])
    kdt = KDTree(atompos_array,  metric='euclidean')
    n_neighbours=n_neighbours if n_neighbours is not None else len(
        atompos_df.z)

    dist_arr=np.array([])

    if check_sub_df is not None:
        check_array = check_sub_df.as_matrix(columns = ['x', 'y', 'z'])
        for i in range(0, len(check_sub_df.z)):
            dist, ind = kdt.query([check_array[i]], k=(n_neighbours+1))
            dist=dist[0]
            dist=dist[dist<nhood_size]
            dist=dist[dist>MIN_dist]
```

```
            dist_arr = np.concatenate((dist_arr, dist[1:]))
    else:
        for i in range(0, len(atompos_df.z)):
            dist, ind = kdt.query([atompos_array[i]], k=(n_neighbours+1))
            dist=dist[0]
            dist=dist[1:]
            dist=dist[dist<nhood_size]
            dist=dist[dist>MIN_dist]
            dist_arr = np.concatenate((dist_arr, dist))

    if plot_hist:
        plt.figure()
        plt.hist(dist_arr, bins=np.arange(np.min(dist_arr), np.max(dist_arr
            ), dbin))

    ####bruke 1D DBSCAN paa dist_arr, returnere  array med means
    X2d = np.zeros((len(dist_arr), 2))
    X2d[:,0] = dist_arr#.reshape(1, -1)

    db = DBSCAN(eps=max_dist, min_samples=1).fit(X2d)

    labels=db.labels_
    # Number of clusters in labels, ignoring noise if present.
    n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

    ##Reknar ut center som CM av kvar cluster..
    layer_list = []
    for i in range(0, n_clusters_):
        layer = X2d[labels==i]
        no_atoms=len(layer[:,0])
        mean = np.mean(layer[:,0])
        std = np.std(layer[:,0])
        dict1 = {'layer_dist_mean': mean, 'layer_dist_std': std, 'no_atoms'
            : no_atoms}
        layer_list.append(dict1)

    layer_db_df = pd.DataFrame(layer_list, columns=['layer_dist_mean', '
        layer_dist_std', 'no_atoms'])

    #add to histogram
    if plot_hist:
        plt.scatter(layer_db_df.layer_dist_mean, np.zeros(len(layer_list)),
             s=20, color='r')

    if plot_kde:
        fig, ax=kde1D(dist_arr)
        if plot_hist:
            ax.scatter(layer_db_df.layer_dist_mean, np.zeros(len(layer_list
                )), s=20, color='r')

    return layer_db_df, dist_arr
```

C.0.0.2 test_data.py

```
"""
```

```
### test_data.py ###

Functions for retrieving data from the folder test_data_folder. Files were
    stored here for each step in the reconstruction process. Current
    directory should be set within the grain_boundary_reconstruction folder
    . If not, following line should be changed in get-functions:
path_dir = os.getcwd()
"""
from skimage import io
import os
import matplotlib.pyplot as plt
from scipy import ndimage
import pandas as pd
import numpy as np

import grain_boundary_reconstruction.tools as tools

#####

def get_original_images():
    '''returns list of arrays containing original images'''
    path_dir = os.getcwd()
    path_images=r'test_data_folder\\original_images_sigma5'
    filename_list = os.listdir(path_dir+'\\'+path_images)
    images_list = [io.imread(path_dir+'\\'+path_images+'\\'+filename,
        as_grey=True) for filename in filename_list]
    return images_list

def get_refined_peaks_original():
    '''returns list of df's containing refined peak positions for averaged
        images.
    This is the starting point for the 3-D reconstruction.'''
    path_dir = os.getcwd()
    path_peaks=r'test_data_folder\\\original_images_atompos'
    filename_list = os.listdir(path_dir+'\\'+path_peaks)
    peaks_list = [pd.DataFrame(np.load(path_dir+'\\'+path_peaks+'\\'+
        filename), columns=['z', 'x_proj']) for filename in filename_list]
    return peaks_list

def get_affined_images():
    '''returns list of arrays containing affined images'''
    path_dir = os.getcwd()
    path_images=r'affined_images'
    images_list=[]
    for i in range(0,5):
        filename='affined_avg_img'+str(i)+'.png'
        image_arr=io.imread(path_dir+'\\'+path_images+'\\'+filename,
            as_grey=True)
        images_list.append(np.array(image_arr, dtype=float))
    return images_list

def get_averaged_images():
    '''returns list of arrays containing averaged images'''
    path_dir = os.getcwd()
    path_images=r'test_data_folder\\averaged_images'
    filename_list = os.listdir(path_dir+'\\'+path_images)
```

```
    images_list = [io.imread(path_dir+'\\'+path_images+'\\'+filename,
        as_grey=True) for filename in filename_list]
    return images_list

def get_refined_peaks_avg():
    '''returns list of df's containing refined peak positions for averaged
        images.
    This is the starting point for the 3-D reconstruction.'''
    path_dir = os.getcwd()
    path_peaks=r'test_data_folder\\averaged_images_atompos'
    filename_list = os.listdir(path_dir+'\\'+path_peaks)
    peaks_list = [pd.DataFrame(np.load(path_dir+'\\'+path_peaks+'\\'+
        filename), columns=['z', 'x_proj']) for filename in filename_list]
    return peaks_list

def get_refined_peaks_avg_scaled():
    '''returns list of df's containing refined peak positions for averaged
        images.
    This is the starting point for the 3-D reconstruction.'''
    path_dir = os.getcwd()
    path_peaks=r'test_data_folder\\averaged_images_atompos_scaled'
    filename_list = os.listdir(path_dir+'\\'+path_peaks)
    peaks_list = [pd.DataFrame(np.load(path_dir+'\\'+path_peaks+'\\'+
        filename), columns=['z', 'x_proj']) for filename in filename_list]
    return peaks_list

def get_reconstructed_from_cif(image_inds, n_units=1):
    '''returns the structure'rec. from image inds from cif-file'''
    images_ind_str = '_images_'
    for ind in image_inds:
        images_ind_str+=(str(ind)+'_')

    path_dir = os.getcwd()
    path_files=r'reconstructions'
    filename_list = os.listdir(path_dir+'\\'+path_files)
    check=0
    #find filename that contains image_inds_str
    filename_start = 'Reconstruction_REAL_'+images_ind_str+'atomradius_'
    for filename in filename_list:
        if filename[0:len(filename_start)]==filename_start:
            if filename[-3:]=='cif':
                df_cif = tools.readCIF(path_dir+'\\'+path_files+'\\'+
                    filename[:-4], n_units=n_units)
                check=1
    if check==0:
        print('File not found!')
        df_cif=0
    return df_cif

def get_reconstructed_from_xyz(image_inds):
    '''returns the structure'rec. from image inds from cif-file'''
    images_ind_str = '_images_'
    for ind in image_inds:
        images_ind_str+=(str(ind)+'_')
    path_dir = os.getcwd()
    path_files=r'reconstructions'
    filename_list = os.listdir(path_dir+'\\'+path_files)
```

```
    check=0
    #find filename that contains image_inds_str
    filename_start = 'Reconstruction_REAL_'+images_ind_str+'atomradius_'
    for filename in filename_list:
        if filename[0:len(filename_start)]==filename_start:
            if filename[-3:]=='xyz':
                df_from_xyz = tools.readXYZ(path_dir+'\\'+path_files+'\\'+
                    filename[:-4])
                check=1
    if check==0:
        print('File not found!')
    return df_from_xyz

def get_ideal_bulk():
    '''Returns ideal bulk'''
    pd.options.mode.chained_assignment = None  # default='warn' #get rid of
        warning
    path=r'test_data_folder\\ideal_3D_structure'
    filename_ideal='half_sigma5.xyz'
    ideal_df = pd.read_csv(path+'\\'+filename_ideal, sep=" ", header=1,
        names = ['atom', 'x', 'y', 'z'])
    ideal_out = ideal_df[['x', 'y', 'z']]
    return ideal_out

def get_reconstructed_structure(n_units=3):
    '''Returns reconstructed structure as multiples of real clustered CIF-
        file...'''
    filename='real_rec_'+str(n_units)+'units.xyz'
    path='test_data_folder'
    return pd.read_csv(path+'\\'+filename, delimiter=' ', header=2, names=[
        'atom', 'x', 'y', 'z'])

def get_angles_deg():
    '''Calculate rotation angles from lattice directions'''
    dirZone =  np.array([0, 0, 1])
    dir38 =  np.array([1, -3, 4])
    dir58 =  np.array([1, -3, 2])
    dir90 =  np.array([1, -3, 0])
    theta38 = np.rad2deg(np.arccos( (np.sum(dirZone*dir38))/( np.sqrt(np.
        sum(dirZone**2))* np.sqrt(np.sum(dir38**2)) ) ))
    theta58 = np.rad2deg(np.arccos( (np.sum(dirZone*dir58))/( np.sqrt(np.
        sum(dirZone**2))* np.sqrt(np.sum(dir58**2)) ) ))
    angles_deg=[0.0, theta38, theta58, 90.0]
    return angles_deg

def get_EDX_images():
    '''returns two lists of arrays containing original EDX images,
    Zr and Y'''
    path_images=r'original_EDX_images_sigma5' ##should be in
        grain_boundar_reconstruction folder!!!
    names_Zr =['zone_Zr K', '38_Zr K', '58_Zr K', '90_Zr K']
    names_Y =['zone_Y K', '38_Y K',  '58_Y K',  '90_Y K']
    EDX_df_list_Zr = []
    EDX_df_list_Y = []
    for i in range(0,len(names_Zr)):
        name=names_Zr[i]
```

```
            EDX_df =  pd.read_csv(path_images+'\\'+name+'.csv', sep=",", header
                =None)
            EDX_df_list_Zr.append(EDX_df)
            name=names_Y[i]
            EDX_df =  pd.read_csv(path_images+'\\'+name+'.csv', sep=",", header
                =None)
            EDX_df_list_Y.append(EDX_df)
        return EDX_df_list_Zr, EDX_df_list_Y

    def plot_EDX_subplot(filter_factor=0):
        EDX_df_list_Zr, EDX_df_list_Y = get_EDX_images()

        fig, axs = plt.subplots(4,3, figsize=(15, 6))
        fig.subplots_adjust(top=0.97,
                            bottom=0.0,
                            left=0.0,
                            right=1.0,
                            hspace=0.085,
                            wspace=0.0)
        X,Y = np.meshgrid(range(0,EDX_df_list_Zr[0].shape[0]),range(0,
            EDX_df_list_Zr[0].shape[1]))
        axs = axs.ravel()
        ind=-1
        for i in [0,3,6,9]:
            ind+=1
            EDX_df_both = EDX_df_list_Zr[ind] + EDX_df_list_Y[ind]
            intensity_both = ndimage.filters.gaussian_filter(np.asarray(
                EDX_df_both), filter_factor)
            axs[i].pcolormesh(X,Y,intensity_both)#, cmap='Greys_r', vmin=vmin,
                vmax=vmax)
            axs[i].set_title("Sum Y and Zr image "+str(ind))
            axs[i].axis('equal')
            axs[i].axis('off')
            EDX_df_Zr = EDX_df_list_Zr[ind]
            intensity_Zr = ndimage.filters.gaussian_filter(np.asarray(EDX_df_Zr
                ), filter_factor)
            axs[i+1].pcolormesh(X,Y,intensity_Zr)#, cmap='Greys_r', vmin=vmin,
                vmax=vmax)
            axs[i+1].set_title("Zr image "+str(ind))
            axs[i+1].axis('equal')
            axs[i+1].axis('off')
            EDX_df_Y = EDX_df_list_Y[ind]
            intensity_Y = ndimage.filters.gaussian_filter(np.asarray(EDX_df_Y),
                 filter_factor)
            axs[i+2].pcolormesh(X,Y,intensity_Y)#, cmap='Greys_r', vmin=vmin,
                vmax=vmax)
            axs[i+2].set_title("Y image "+str(ind))
            axs[i+2].axis('equal')
            axs[i+2].axis('off')

    def get_visual_guess_GB_z_list(average=False):
        '''returns list of visual GB positions'''
        if average:
            ut = np.array([[194, 214],[405, 450], [467, 511], [248, 262]])
        else:
            ut = np.array([[198, 214],[405, 450], [473, 520], [248, 262]])
        return ut
```

```
def refine_GB_z_values_and_find_x_proj_unit_dist(atompos_df_list,
    visual_guess_GB_z_list, dz=5, plot=False):
    '''Takes initial guess for GB position, returns mean z-values of the
        atom layers indicated'''

    visual_guess_GB_z_list_refined = []
    unit_x_proj_dist_list=[]
    for i in range(0,4):
        pos_df = atompos_df_list[i]
        GB_L_z = visual_guess_GB_z_list[i][0]
        GB_R_z = visual_guess_GB_z_list[i][1]
        GB_L_inds =[np.array(np.array([pos_df.z<(GB_L_z+dz) ])*np.array([
            pos_df.z>(GB_L_z-dz) ]))][0][0]
        GB_R_inds =[np.array(np.array([pos_df.z<(GB_R_z+dz) ])*np.array([
            pos_df.z>(GB_R_z-dz) ]))][0][0]
        GB_L_z_refined = np.mean(pos_df.z[GB_L_inds])
        GB_R_z_refined = np.mean(pos_df.z[GB_R_inds])

        visual_guess_GB_z_list_refined.append([GB_L_z_refined,
            GB_R_z_refined])
        if plot:
            plt.figure()
            plt.scatter(pos_df.z, pos_df.x_proj,  s=10, color='b', label='
                bulk atoms')
            plt.scatter(pos_df.z[GB_L_inds], pos_df.x_proj[GB_L_inds],  s
                =4, color='r', label='left GB layer')
            plt.scatter(pos_df.z[GB_R_inds], pos_df.x_proj[GB_R_inds],  s
                =4, color='y', label='right GB layer')
            plt.axis('equal')
            plt.legend()

        #Find distance between GB units
        x_vals = sorted(np.array(pos_df.x_proj[GB_L_inds]))

        if len(x_vals)<2:
            print('Too few GB atoms found')
        else:
            GB_unit_2_atom_x=x_vals[0]
            if i==0:
                GB_unit_1_atom_x=x_vals[2]
            if i==1:
                GB_unit_1_atom_x=x_vals[1]
            if i==2:
                GB_unit_1_atom_x=x_vals[1]
            if i==3:
                GB_unit_1_atom_x=x_vals[3]

            unit_x_proj_dist = abs(GB_unit_2_atom_x-GB_unit_1_atom_x)
            unit_x_proj_dist_list.append(unit_x_proj_dist)
    return visual_guess_GB_z_list_refined, unit_x_proj_dist_list

def get_scale_factors():
    '''returns scale factors (px-to-angstrom ratio) for affined images (
        should be same as for averaged images). '''
    path_dir = os.getcwd()
    path=r'test_data_folder'
```

```
    scale_factors = np.load(path_dir+'\\'+path+'\\scale_factors_avg.npy')
    return scale_factors
```

C.0.0.3 peakFinder.py

```
"""
### peakFinder.py ###

Functions for finding intensity peaks and fitting 2-D Gaussian function to
    the data
"""
import pandas as pd
import numpy as np
from scipy import optimize

#####

def find_maxima(image, lower_intensity_limit_factor, minimum_distance):
    '''return all maxima above lower_intensity_limit_factor(factor of image
         max), IF distance bigger than min_dist
    atoms clores to the edge than 1 atom radius are ignored'''

    #add zero padding (2*atomradius) to remove problem with atoms at edges
    image_copy = np.zeros(((image.shape[0] + minimum_distance*4), (image.
        shape[1] + minimum_distance*4)))
    image_copy[minimum_distance*2:-minimum_distance*2, minimum_distance*2:-
        minimum_distance*2] += image
    max_list=[]
    lower_intensity_limit = lower_intensity_limit_factor*np.max(image)
    y_size, x_size=image.shape

    peak_found=True
    while peak_found==True:
        if np.max(image_copy)>lower_intensity_limit:
            y_max, x_max = np.unravel_index(np.argmax(image_copy),
                image_copy.shape) # find the match
            ##Brute force: ser area around peak to zero, find next max
            image_copy[y_max-minimum_distance:y_max+minimum_distance, x_max
                -minimum_distance:x_max+minimum_distance ] = 0
            #remove padding in saved coordinate
            dict1 = {'z': x_max-minimum_distance*2, 'x_proj': y_max-
                minimum_distance*2 }
            #if not on edge, add to list:
            if dict1['z']>minimum_distance and dict1['z']<x_size-
                minimum_distance:
                if dict1['x_proj']>minimum_distance and dict1['x_proj']<
                    y_size-minimum_distance:
                    max_list.append(dict1)
        else:
            peak_found=False

    maxima_df = pd.DataFrame(max_list, columns=['z', 'x_proj'])
    print("Number of atoms detected: "+str((len(maxima_df.z))))
    print("Intensity level minimum: "+str(lower_intensity_limit_factor*100)
        +" % of max intensity")
```

```
    return maxima_df

def gaussian(amplitude, mu_x, mu_y, sigma_x, sigma_y):
    """Returns a gaussian function with the given parameters"""
    return lambda x,y: amplitude*np.exp(
                -(((mu_x-x)/sigma_x)**2+((mu_y-y)/sigma_y)**2)/2)

def moments(data):
    """Returns (amplitude, x, y, sigma_x, sigma_y)
    the gaussian parameters of a 2D distribution by calculating its
    moments """
    total = data.sum()
    X, Y = np.indices(data.shape)
    x = (X*data).sum()/total
    y = (Y*data).sum()/total
    col = data[:, int(y)]
    sigma_x = np.sqrt(np.abs((np.arange(col.size)-y)**2*col).sum()/col.sum
        ())
    row = data[int(x), :]
    sigma_y = np.sqrt(np.abs((np.arange(row.size)-x)**2*row).sum()/row.sum
        ())
    amplitude = data.max()
    return amplitude, x, y, sigma_x, sigma_y

def fitgaussian(data):
    """Returns (amplitude, x, y, sigma_x, sigma_y)
    the gaussian parameters of a 2D distribution found by least-squares fit
        """
    params = moments(data)
    errorfunc = lambda p: np.ravel(gaussian(*p)(*np.indices(data.shape)) -
        data)
    params_bestfit, success = optimize.leastsq(errorfunc, params)
    return params_bestfit

def fit_gaussians_to_maxima(image, initial_peaks_df, atomradius, iterations
    =1, intensity=False):
    '''Gaussian fit to square box of the image of size 2*atomradius around
        each initial peak position'''
    #add padding to avoid problems at edges
    image_copy = np.zeros(((image.shape[0] + atomradius*4), (image.shape[1]
         + atomradius*4)))
    image_copy[atomradius*2:-atomradius*2, atomradius*2:-atomradius*2] +=
        image
    positions_refined_list=[]
    y_size, x_size=image.shape

    move_list=[]
    for ind, x,y in zip(initial_peaks_df.index,initial_peaks_df.z,
        initial_peaks_df.x_proj):
        #choose area (atomradius)x(atomradius) around each peak for
            gaussian fitting
        atom_snip = image_copy[int(y+atomradius):int(y+atomradius*3), int(x
            +atomradius):int(x+atomradius*3)].copy()
        params = fitgaussian(atom_snip)
        (amplitude, y_gauss, x_gauss, sigma_x, sigma_y) = params
        move= np.sqrt((y_gauss-atomradius)**2 + (x_gauss-atomradius)**2)
```

```
            if move<5: #ignore those atoms that go very far(jumps out of image)
                move_list.append(move)
            if intensity:
                dict1 = {'z': x_gauss-atomradius+x, 'x_proj': y_gauss-
                    atomradius+y, 'intensity': amplitude } #remove padding
            else:
                dict1 = {'z': x_gauss-atomradius+x, 'x_proj': y_gauss-
                    atomradius+y } #remove padding
            #if not on edge, add to list:
            if dict1['z']>atomradius and dict1['z']<x_size-atomradius:
                if dict1['x_proj']>atomradius and dict1['x_proj']<y_size-
                    atomradius:
                    positions_refined_list.append(dict1)
        if intensity:
            maxima_refined_df = pd.DataFrame(positions_refined_list, columns=['
                z', 'x_proj', 'intensity'])
        else:
            maxima_refined_df = pd.DataFrame(positions_refined_list, columns=['
                z', 'x_proj'])
        print("Number of atoms detected: "+str((len(maxima_refined_df.z))))
        print("Average position move: "+str(np.mean(move_list)))
        #return maxima_refined_CM_df
        for it in range(iterations-1):
            maxima_refined_df=fit_gaussians_to_maxima(image, maxima_refined_df,
                 atomradius, iterations=1, intensity=intensity)
        return maxima_refined_df
```

C.0.0.4 geometric_correction.py

```
"""
### geometric_correction.py ###

Functions for correction of image distortion by estimating an affime
    transformation from an FCC reference lattice (template)
"""
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.neighbors import KDTree
import random
from skimage import transform

import grain_boundary_reconstruction.tools as tools

#####

def create_bulk_templates(ideal_bulk_df, projection_angles_deg, side='left'
    ):
    '''returns list of dataframes with templates created from projections
        of the ideal bulk structure.
    GB plane set to 0 on z-axis.'''
    template_list=[]
    for proj_angle in projection_angles_deg:
        ideal_bulk_df_proj = tools.find_projected_coordinates(ideal_bulk_df
            , proj_angle)
```

```
        max_dist=0.1
        ideal_bulk_df_proj_clustered = tools.cluster_2D_projection(
            ideal_bulk_df_proj, max_dist)

        ##Shift GB to 0
        if side=='left':
            ideal_bulk_df_proj_clustered.z *= -1
            ideal_bulk_df_proj_clustered.z -= np.max(
                ideal_bulk_df_proj_clustered.z)
        if side=='right':
            ideal_bulk_df_proj_clustered.z -= np.min(
                ideal_bulk_df_proj_clustered.z)

        template_list.append(ideal_bulk_df_proj_clustered)
    return template_list

def find_layer(refined_peaks_df, side, dz=5):
    '''returns df with layer atoms, at least 3 stk'''
    if side=='left':
        z_layer_image_df = refined_peaks_df.ix[refined_peaks_df.z<(np.min(
            refined_peaks_df.z)+dz)]
        it=0
        while len(z_layer_image_df)<3:#test next layer
            it+=1
            z_layer_image_df = refined_peaks_df.ix[refined_peaks_df.z<(np.
                min(refined_peaks_df.z)+it*dz)]
            z_layer_image_df = z_layer_image_df.ix[z_layer_image_df.z>=(np.
                min(z_layer_image_df.z)+(it-1)*dz)]

    if side=='right':
        z_layer_image_df = refined_peaks_df.ix[refined_peaks_df.z>(np.max(
            refined_peaks_df.z)-dz/2)]
        it=0
        while len(z_layer_image_df)<3:#test next layer
            it+=1
            z_layer_image_df = refined_peaks_df.ix[refined_peaks_df.z>(np.
                max(refined_peaks_df.z)-it*dz)]
            z_layer_image_df = z_layer_image_df.ix[z_layer_image_df.z<=(np.
                max(z_layer_image_df.z)-(it-1)*dz)]
    return z_layer_image_df

def match_initial_points(template_edge_df, z_layer_image_df, match_atoms=
    None, atoms_im=None, atoms_temp=None):
    '''match_atoms: list of numbers in template to match with first atoms
        in image.
    if 'all', no gaps in image atoms, all atoms used'''
    sorted_im = z_layer_image_df.sort_values('x_proj').copy()
    sorted_im.index=np.arange(0, len(sorted_im.index))
    sorted_tem = template_edge_df.sort_values('x_proj').copy()
    sorted_tem.index=np.arange(0, len(sorted_tem.index))

    if  match_atoms=='all':
        shortest=len(sorted_tem.index) if len(sorted_tem.index)<=len(
            sorted_im.index) else len(sorted_im.index)
        atoms_im = np.arange(0, shortest)
        atoms_temp = np.arange(0, shortest)
```

```
    im_points = sorted_im.ix[np.array(atoms_im)].copy()
    tem_points = sorted_tem.ix[np.array(atoms_temp)].copy()
    scale_factor_x_proj = abs(np.max(im_points.x_proj)-np.min(im_points.
        x_proj))/abs(np.max(tem_points.x_proj)-np.min(tem_points.x_proj))
    scaled_tem = tem_points*scale_factor_x_proj

    translation_x_proj = np.min(im_points.x_proj)-np.min(scaled_tem.x_proj)
    translation_z = im_points.z[np.argmin(im_points.x_proj)] - scaled_tem.z
        [np.argmin(scaled_tem.x_proj)]

    return scale_factor_x_proj, translation_x_proj, translation_z


def match_two_point_sets(template_df, image_df, min_dist=None, plot=False):
    '''Point pair: one in template_df and one in image_df CLOSER than
        smallest_dist/2 (smallest dist within template)
    returns dfs matched by index. Template input must be scaled and matched
        .'''
    template_array = template_df.as_matrix(columns = ['x_proj', 'z'])
    kdt = KDTree(template_array,  metric='euclidean')
    dist, inds = kdt.query([[template_array[0][0],template_array[0][1]]], k
        =3)

    #smallest distance within template: Use half this as limit for NN
    smallest_dist = min_dist if min_dist is not None else dist[0][1]

    image_array = image_df.as_matrix(columns = ['x_proj', 'z'])
    kdt = KDTree(image_array,  metric='euclidean')

    #iterate over all atoms in template. find nearest neighbour in the
        image peaks. IF clore enough, add to list.
    template_points_with_neighbour=[]
    image_points_with_neighbour=[]
    for atom_ind in template_df.index:
            dist, inds = kdt.query([[template_df.x_proj[atom_ind],
                template_df.z[atom_ind]]], k=1)

            if dist[0][0]<smallest_dist/2:
                dict_template = {'z': template_df.z[atom_ind], 'x_proj':
                    template_df.x_proj[atom_ind] }
                dict_image = {'z': image_array[inds[0][0]][1], 'x_proj':
                    image_array[inds[0][0]][0]}

                template_points_with_neighbour.append(dict_template)
                image_points_with_neighbour.append(dict_image)

    template_matched_df = pd.DataFrame(template_points_with_neighbour,
        columns=['z', 'x_proj'])
    image_matched_df = pd.DataFrame(image_points_with_neighbour, columns=['
        z', 'x_proj'])

    colors = [plt.cm.Spectral(each)
              for each in np.linspace(0, 1, len(image_matched_df.z))]
    random.shuffle(colors)

    if plot:
        fig, ax = plt.subplots()
```

```
        for i in template_matched_df.index:

            ax.scatter(template_matched_df.z[i], template_matched_df.x_proj
                [i], color=tuple(colors[i]))
            ax.scatter(image_matched_df.z[i], image_matched_df.x_proj[i],
                color=tuple(colors[i]))
        plt.xlabel('z')
        plt.ylabel('x_proj')
        ax.axis('equal')
        ax.set_ylim(ax.get_ylim()[::-1])

    return template_matched_df, image_matched_df

def get_coeffinients(affine_matrix):
    '''calculated coefficients from affine matrix:

    A = np.arrar([[a0, a1, a2],
                  [b0, b1, b2]])
    translation vector t = [a2, b2] = [tx, ty]
    a0=sx cos(T)
    a1=-sy sin(T + C) #C is shear angle, T is rotation angle
    b0 = sx sin(T)
    b1 = sy sin(T + C)
    '''
    [a0, a1, a2, b0, b1, b2] =  affine_matrix.ravel()[0:6]

    #translation
    tx, ty = a2, b2
    #scaling
    sx = np.sqrt(a0**2 + b0**2)
    #rotation angle
    T = np.arctan(b0/a0)
    #shear angle
    C = np.arctan(-a1/b1) -T
    sy = -a1/np.sin(T+C) #=b1/np.cos(T+C)

    print('Rotation angle T: %f degrees \n'%(np.rad2deg(T)))
    print('Shear angle C (x-shear): %f degrees \n'%(np.rad2deg(C)))
    print('Scale factors [sx, sy]: \n [%f, %f] \n'%(sx, sy))
    print('Translation vector [tx, ty]: \n [%f, %f] \n'%(tx, ty))
    print('Returns [sx, sy, T, C, tx, ty] \n ')
    return [sx, sy, T, C, tx, ty]

####Perform affine transformation

def estimate_affine_matrix(template_matched_df,image_matched_df):
    '''estimate affine transformation matrix. Returns matrix as numpy array
        '''
    src = template_matched_df.as_matrix(['z', 'x_proj'])
    dst = image_matched_df.as_matrix(['z', 'x_proj'])
    tform = transform.estimate_transform('affine', src, dst) #same! #
        handlar vel berre om aa faa ut rett type matrix
    print('Transformation matrix: \n')
    print(tform.params)
    affine_matrix=np.array(tform.params)
    return affine_matrix
```

```
def apply_affine_transform(affine_matrix, image):
    '''takes image as array, matrix as array, applies transformation from
        matrix, returns image as array'''
    tform = transform.AffineTransform(affine_matrix)
    # Normalize image to between 0 and 255
    image_for_aff = ((image-image.min())/(image.max()-image.min()))*225
    warped = transform.warp(image_for_aff, tform, preserve_range=True)#,
        output_shape=(50, 300))
    return warped
```

C.0.0.5 noise_reduction.py

```
"""
### noise_reduction.py ###

Functions for image averaging by cross-correlation to reduce noise
"""
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from skimage.feature import match_template
from scipy.signal import argrelextrema
from statsmodels.nonparametric.smoothers_lowess import lowess

import grain_boundary_reconstruction.tools as tools

#####

def generate_template(image, template_height=100):
    '''creates image_snip = image without edges
    creates template covering full width'''
    template_start_x=0  # change to ignore edges
    template_start_y=0  # change to ignore edges
    template_size=[image.shape[1]-2*template_start_x, template_height]
    snip_size=[template_size[0], image.shape[0]]

    #create template
    template=image[template_start_y:(template_start_y+template_size[1]),
        template_start_x:(template_start_x+template_size[0])] ##using
        almost full width, only 1 GB unit height
    ##create image with same width as template, full height #remove edges
    image_snip=image[0:snip_size[1], template_start_x:(template_start_x+
        template_size[0])]
    return image_snip, template

def cross_correlation_along_GB(image, plot=True, template_height=100,
    scatter_peaks=None):
    '''Creates a template within image, shifts if vertically across the
        image and calculates cross-correlation function (CCF).
    returns indices for maxima'''
    image_snip, template = generate_template(image, template_height=
        template_height)
    CCF_GB = match_template(image_snip, template)
    #filtering to easier find peaks
```

```
    CCF_GB_filtered = lowess(CCF_GB[:,0], range(0, np.shape(CCF_GB[:,0])[0]
        ),frac = 0.04)[:,1]# frac=im_df.signal_filtering_frac[ind])[:,1]
    max_ind = argrelextrema(np.array(CCF_GB_filtered), np.greater)[0]

    if plot:
        fig,ax =tools.plot_image_array(image_snip)
        #frame template
        rect = patches.Rectangle((1,1),image_snip.shape[1]-3,
            template_height,linewidth=2,edgecolor='r',facecolor='none')
        ax.add_patch(rect)
        plt.figure()
        plt.plot(CCF_GB_filtered, color='k', linewidth=0.5)
        plt.scatter(max_ind, [CCF_GB_filtered[ind] for ind in max_ind],
            color='b')
        if scatter_peaks is not None:
            plt.scatter(max_ind[scatter_peaks], [CCF_GB_filtered[ind] for
                ind in max_ind[scatter_peaks]], color='r', marker='*')
    return max_ind

def create_averaged_image(image, shifts, averaged_image_height):
    '''returns image averaged over units at image positions indicated by
        shifts'''
    image_snip, template = generate_template(image)
    image_averaged=np.zeros((averaged_image_height, image_snip.shape[1]))
    no_units=1
    for shift in shifts:
        if shift+averaged_image_height<image_snip.shape[0]: #ensure that
            indices are wiithin image
            template_shifted=image_snip[shift:shift+averaged_image_height,
                :]
            image_averaged+=template_shifted
            no_units+=1
    print('Image averaged over %d units'%no_units)
    return image_averaged, no_units
```

C.0.0.6 reconstruction.py

```
"""
### reconstruction.py ###

Functions for 3-D reconstruction from lines and image alignment
"""
import pandas as pd
import numpy as np
from sklearn.neighbors import KDTree
import matplotlib.pyplot as plt

import grain_boundary_reconstruction.tools as tools

#####

def find_atoms_from_lines(im1pos_df, im2pos_df, angles, atomradius):
    '''Takes two df med attributes x_proj, z, one for each image (ferdig
        aligna), and finds intersections..
```

```
    "atomradius": overlap if distance between liner are less than 2*
        atomradius
    angles=[a1, a2]'''

    A = np.array([[np.cos(np.deg2rad(angles[0])),  np.sin(np.deg2rad(angles
        [0]))],
         [np.cos(np.deg2rad(angles[1])), np.sin(np.deg2rad(angles[1]))]])
    #deg2rad gir rounding errors, setter derfor lave tal til 0 her
    A[A < 7e-17] = 0


    coords_3D_list=[]
    for i in im1pos_df.index:#range(0, len(im1pos_df.z)):
        for j in im2pos_df.index:#range(0, len(im2pos_df.z)):
            z_dist = np.abs(im1pos_df.z[i]-im2pos_df.z[j])

            if z_dist <2*atomradius: #overlap
                z_out = (im1pos_df.z[i]+im2pos_df.z[j])/2
                w = np.array([im1pos_df.x_proj[i], im2pos_df.x_proj[j]])
                v = np.linalg.solve(A,w)
                dict1 = {'atom': 'Zr', 'x': v[0], 'y': v[1], 'z': z_out}
                coords_3D_list.append(dict1)

    pos_3D_df = pd.DataFrame(coords_3D_list, columns=['atom','x','y','z'])
    return pos_3D_df

def find_atoms_from_lines_many_images(atompos_df_list, angle_list,
    atomradius, max_dist_intersections=None):
    '''Takes list of MINIMUM 2  df med attributes x_proj, z, one for each
        image (ferdig aligna), and finds intersections..
    "atomradius": overlap if distance between liner are less than 2*
        atomradius
    angles=[a1, a2, a3]
    Uses e.find_atoms_from_lines and e.cluster '''
    max_dist_intersections = max_dist_intersections if
        max_dist_intersections is not None else atomradius

    no_df = len(atompos_df_list) #minimum2 !

    pairs_list = []

    #i,j=[0,1]
    ##no_df = 2
    for i in range(0, no_df):
        for j in range(i, no_df):
            if i!=j:
                print('Images combined: '+str(i)+' and '+str(j))

                pos_3D_df_pair = find_atoms_from_lines(atompos_df_list[i],
                    atompos_df_list[j], [angle_list[i],angle_list[j]],
                    atomradius)
                print('Number of atoms found: '+str(len(pos_3D_df_pair)))
                pairs_list.append(pos_3D_df_pair)

    #combine
    all_df = pd.concat(pairs_list)
```

```
    #3D clustering
    if no_df>2:
        minsample=no_df #cluster: find mean if three intersections are
            close...
        pos_3D_df = tools.cluster_3D(all_df, minsample, max_dist=
            max_dist_intersections, maxsize=no_df)
        #pos_3D_df = cluster(all_df, minsample, max_dist=
            max_dist_intersections)
        #tools.plot3Dstructure(pos_3D_df)
    else:
        pos_3D_df=all_df
    return pos_3D_df

def align_image_to_reprojection(reproj_cluster_df, image_df, interval, dx,
    d_NN_proj_row, plot=True):
    '''Finds how much image_df should be moved to minimize mean dist to
        reprojected image atoms'''
    #check within each layer only. and only if nn is less than dv/2 to
        ensure that not next row nn is used (if atoms are missing)

    ##IF we make sure the interval is less than d_NN_proj: can use NN in
        combined image with both reproj and image_adjust!
    #for each atom in image+adjust, check if it has NN

    image_adjusted_df = image_df.copy()
    image_adjusted_df_bulk = pd.concat([image_adjusted_df[image_adjusted_df
        .z<-0.5], image_adjusted_df[image_adjusted_df.z>2.5]])

    image_adjusted_df_bulk.x_proj+=interval[0] #set image to first test
        position

    atompos_array = reproj_cluster_df.as_matrix(columns = ['x_proj', 'z'])

    mean_list=[]
    for adj in np.arange(interval[0], interval[1], dx):
        image_adjusted_df_bulk.x_proj+=dx #move image slowly upwards

        NN_dist_list=[]
        for atom_ind in image_adjusted_df_bulk.index:
            kdt = KDTree(atompos_array,  metric='euclidean')
            dist, inds = kdt.query([[image_adjusted_df_bulk.x_proj[atom_ind
                ],image_adjusted_df_bulk.z[atom_ind]]], k=1)
            #dist, inds = kdt.query([[0,0]], k=1)

            NN_dist_list.append(dist[0][0])
        NN_dist_array=np.array(NN_dist_list)
        mean_list.append(np.mean(NN_dist_array[NN_dist_array<d_NN_proj_row
            ])) #just to remove the edge atoms...

    if plot:
        plt.figure()
        plt.scatter(np.arange(interval[0], interval[1], dx),mean_list)

    minimizing_adj = np.arange(interval[0], interval[1], dx)[np.argmin(
        mean_list)]
```

```
        print('Image shift minimizing mean distance to nearest neighbour: %.2f'
            %(minimizing_adj))
        print('Mean distance to nearest neighbour: %.2f'%(np.min(mean_list)))

        if plot:
            plt.figure()
            plt.scatter(reproj_cluster_df.z, reproj_cluster_df.x_proj, s=10,
                color='b')
            plt.scatter(image_adjusted_df.z, image_adjusted_df.x_proj+
                minimizing_adj, s=5, color='r')
            plt.title('Image shifted %.2f vertically'%(minimizing_adj))
        return minimizing_adj
```

Bibliography

[1] D. Hull and D. J. Bacon, *Introduction to Dislocations*. Elsevier, 5 ed., 2011.

[2] J. P. Buban, K. Matsunaga, J. Chen, N. Shibata, W. Y. Ching, T. Yamamoto, and Y. Ikuhara, "Grain boundary strengthening in alumina by rare earth impurities," *Science*, vol. 311, no. 5758, pp. 212–215, 2006.

[3] W. Lee, H. J. Jung, M. H. Lee, Y. B. Kim, J. S. Park, R. Sinclair, and F. B. Prinz, "Oxygen Surface Exchange at Grain Boundaries of Oxide Ion Conductors," *Advanced Functional Materials*, vol. 22, no. 5, pp. 965–971, 2012.

[4] R. F. Klie, J. P. Buban, M. Varela, A. Franceschetti, C. Jooss, Y. Zhu, N. D. Browning, S. T. Pantelides, and S. J. Pennycook, "Enhanced current transport at grain boundaries in high-T c superconductors," *Nature*, vol. 435, no. 7041, pp. 475–478, 2005.

[5] X. Guo and R. Waser, "Electrical properties of the grain boundaries of oxygen ion conductors: Acceptor-doped zirconia and ceria," *Progress in Materials Science*, vol. 51, no. 2, pp. 151–210, 2006.

[6] T. Watanabe, "Grain boundary engineering: Historical perspective and future prospects," *Journal of Materials Science*, vol. 46, no. 12, pp. 4095–4115, 2011.

[7] L. Lambert and T. Mulvey, "Ernst Ruska (1906–1988), Designer Extraordinaire of the Electron Microscope: A Memoir," *Advances in Imaging and Electron Physics*, vol. 95, no. C, pp. 2–62, 1996.

[8] S. J. Pennycook and P. D. Nellist, *ScanningTransmission Electron Microscopy — imaging and analysis*. Springer, 2011.

[9] S. Morishita, R. Ishikawa, Y. Kohno, H. Sawada, N. Shibata, and Y. Ikuhara, "Attainment of 40.5 pm spatial resolution using 300 kV scanning transmission electron microscope equipped with fifth-order aberration corrector," *Microscopy*, vol. 67, no. 1, pp. 46–50, 2018.

[10] J. Miao, P. Ercius, and S. J. L. Billinge, "Atomic electron tomography: 3D structures without crystals," *Science*, vol. 353, no. 6306, 2016.

[11] M. C. Scott, C. C. Chen, M. Mecklenburg, C. Zhu, R. Xu, P. Ercius, U. Dahmen, B. C. Regan, and J. Miao, "Electron tomography at 2.4-ångström resolution," *Nature*, vol. 483, no. 7390, pp. 444–447, 2012.

[12] Z. Chen, D. Taplin, M. Weyland, L. J. Allen, and S. Findlay, "Composition measurement in substitutionally disordered materials by atomic resolution energy dispersive X-ray spectroscopy in scanning transmission electron microscopy," *Ultramicroscopy*, vol. 176, no. July 2016, pp. 52–62, 2017.

[13] R. Xu, C. C. Chen, L. Wu, M. C. Scott, W. Theis, C. Ophus, M. Bartels, Y. Yang, H. Ramezani-Dakhel, M. R. Sawaya, H. Heinz, L. D. Marks, P. Ercius, and J. Miao, "Three-dimensional coordinates of individual atoms in materials revealed by electron tomography," *Nature Materials*, vol. 14, no. 11, pp. 1099–1103, 2015.

[14] B. Goris, S. Bals, W. den Broek, E. Carbó-Argibay, S. Gómez-Graña, L. M. Liz-Marzán, and G. Van Tendeloo, "Atomic-scale determination of surface facets in gold nanorods," *Nature Materials*, vol. 11, 2012.

[15] Y. Yang, C.-C. C. Chen, M. C. Scott, C. Ophus, R. Xu, A. Pryor, L. Wu, F. Sun, W. Theis, J. Zhou, M. Eisenbach, P. R. C. Kent, R. F. Sabirianov, H. Zeng, P. Ercius, and J. Miao, "Deciphering chemical order/disorder and material properties at the single-atom level," *Nature*, 2017.

[16] R. G. Parr, "Density Functional Theory of Atoms and Molecules," in *Horizons of Quantum Chemistry*, p. 352, Oxford University Press, 1980.

[17] W. Bulatov, Vasily and Cai, *Computer Simulations of Dislocations*. Oxford University Press, Inc, 2006.

[18] R. M. Ormerod, "Solid oxide fuel cells," *Chemical Society Reviews*, vol. 32, no. 1, pp. 17–28, 2003.

[19] S. J. Hao, C. Wang, T. L. Liu, Z. Q. Z. M. Mao, Z. Q. Z. M. Mao, and J. L. Wang, "Fabrication of nanoscale yttria stabilized zirconia for solid oxide fuel cell," *International Journal of Hydrogen Energy*, vol. 42, no. 50, pp. 29949–29959, 2017.

[20] M. Aoki, Y. M. Chiang, I. Kosacki, L. J. R. Lee, H. Tuller, and Y. Liu, "Solute segregation and grain-boundary impedance in high-purity stabilized zirconia," *Journal of the American Ceramic Society*, vol. 79, no. 5, pp. 1169–1180, 1996.

[21] M. J. Verkerk, B. J. Middelhuis, and A. J. Burggraaf, "Effect of grain boundaries on the conductivity of high-purity ZrO2Y2O3 ceramics," *Solid State Ionics*, vol. 6, no. 2, pp. 159–170, 1982.

[22] T. Oyama, M. Yoshiya, H. Matsubara, and K. Matsunaga, "Numerical analysis of solute segregation at Σ5 (310) [001] symmetric tilt grain boundaries in Y2O3-doped ZrO2," *Physical Review B - Condensed Matter and Materials Physics*, vol. 71, no. 22, pp. 2–4, 2005.

[23] E. C. Dickey, X. Fan, and S. J. Pennycook, "Structure and Chemistry of Yttria-Stabilized Cubic-Zirconia Symmetric Tilt Grain Boundaries," *Journal of the American Ceramic Society*, vol. 84, no. 6, pp. 1361–1368, 2001.

[24] A. Kelly and K. M. Knowles, *Crystallography and Crystal Defects*. John Wiley & Sons, Ltd, 2 ed., nov 2012.

[25] C. Kittel, *Introduction to solid state physics*. Wiley, 7th ed ed., 1996.

[26] J. P. Hirth, J. Lothe, and T. Mura, "Theory of Dislocations (2nd ed.)," *Journal of Applied Mechanics*, vol. 50, no. 2, p. 476, 1983.

[27] D. M. Owen and A. H. Chokshi, "The high temperature mechanical characteristics of superplastic 3 mol% yttria stabilized zirconia," *Acta Materialia*, vol. 46, no. 2, pp. 667–679, 1998.

[28] S. Kobayashi, T. Maruyama, S. Saito, S. Tsurekawa, and T. Watanabe, "In situ observations of crack propagation and role of grain boundary microstructure in

nickel embrittled by sulfur," *Journal of Materials Science*, vol. 49, no. 11, pp. 4007–4017, 2014.

[29] X. Guo, "Physical origin of the intrinsic grain-boundary resistivity of stabilized-zirconia: Role of the space-charge layers," *Solid State Ionics*, vol. 81, no. 3-4, pp. 235–242, 1995.

[30] S. Terashima, T. Kohno, A. Mizusawa, K. Arai, O. Okada, T. Wakabayashi, M. Tanaka, and K. Tatsumi, "Improvement of thermal fatigue properties of Sn-Ag-Cu lead-free solder interconnects on casio's wafer-level packages based on morphology and grain boundary character," in *Journal of Electronic Materials*, vol. 38, pp. 33–38, 2009.

[31] Y. Ikuhara, "Grain Boundary and Interface Structures in Ceramics.," *Journal of the Ceramic Society of Japan*, vol. 109, no. 1271, pp. S110–S120, 2001.

[32] M. L. Kronberg and F. H. Wilson, "Secondary recrystallization in copper," *Trans. Met. Soc. AIME*, vol. 1947, no. 185, pp. 501–514, 1949.

[33] B. Feng, N. R. Lugg, A. Kumamoto, Y. Ikuhara, and N. Shibata, "Direct Observation of Oxygen Vacancy Distribution across Yttria-Stabilized Zirconia Grain Boundaries," *ACS Nano*, vol. 11, no. 11, pp. 11376–11382, 2017.

[34] N. Shibata, F. Oba, T. Yamamoto, and Y. Ikuhara, "Structure, energy and solute segregation behaviour of [110] symmetric tilt grain boundaries in yttria-stabilized cubic zirconia," *Philosophical Magazine*, vol. 84, no. 23, pp. 2381–2415, 2004.

[35] H. Yanagida, K. Koumoto, and M. Miyayama, *The Chemistry of Ceramics*. Wiley, 1 ed., 1996.

[36] C. Pascual, J. R. Jurado, and P. Duran, "Electrical behaviour of doped-yttria stabilized zirconia ceramic materials," *Journal of Materials Science*, vol. 18, no. 5, pp. 1315–1322, 1983.

[37] F. A. Kröger and H. J. Vink, "Relations between the Concentrations of Imperfections in Crystalline Solids," *Solid State Physics - Advances in Research and Applications*, vol. 3, no. C, pp. 307–435, 1956.

[38] B. Feng, T. Yokoi, A. Kumamoto, M. Yoshiya, Y. Ikuhara, and N. Shibata, "Atomically ordered solute segregation behaviour in an oxide grain boundary," *Nature Communications*, vol. 7, p. 11079, 2016.

[39] M. A. Frechero, M. Rocci, G. Sánchez-Santolino, A. Kumar, J. Salafranca, R. Schmidt, M. R. Díaz-Guillén, O. J. Durá, A. Rivera-Calzada, R. Mishra, S. Jesse, S. T. Pantelides, S. V. Kalinin, M. Varela, S. J. Pennycook, J. Santamaria, and C. Leon, "Paving the way to nanoionics: Atomic origin of barriers for ionic transport through interfaces," *Scientific Reports*, vol. 5, pp. 1–9, 2015.

[40] J. A. Moriarty, V. Vitek, V. V. Bulatov, and S. Yip, "Atomistic simulations of dislocations and defects," in *Journal of Computer-Aided Materials Design*, vol. 9, pp. 99–132, 2002.

[41] K. Binder and B. K. Binder, "Atomistic modeling of materials properties by Monte Carlo Simulation," *Advanced materials*, vol. 4, no. 9, pp. 540–547, 1992.

[42] T. Yokoi, M. Yoshiya, and H. Yasuda, "Nonrandom point defect configurations and driving force transitions for grain boundary segregation in trivalent cation doped ZrO22," *Langmuir*, vol. 30, no. 47, pp. 14179–14188, 2014.

[43] H. B. Lee, F. B. Prinz, and W. Cai, "Atomistic simulations of grain boundary segregation in nanocrystalline yttria-stabilized zirconia and gadolinia-doped ceria solid oxide electrolytes," *Acta Materialia*, vol. 61, no. 10, pp. 3872–3887, 2013.

[44] H. B. Lee, F. B. Prinz, and W. Cai, "Atomistic simulations of surface segregation of defects in solid oxide electrolytes," *Acta Materialia*, vol. 58, no. 6, pp. 2197–2206, 2010.

[45] D. B. Williams and C. B. Carter, *Transmission Electron Microscopy; A Textbook for Materials Science*, vol. V1-V4. Springer ScienceþBusiness Media, New York, 2009.

[46] S. J. Pennycook, "SCANNING TRANSMISSION ELECTRON MICROSCOPY : Z - CONTRAST IMAGING," in *Characterization of Materials* (Elton N. Kaufmann, ed.), pp. 1736–1763, John Wiley & Sons, Inc., 2012.

[47] A. V. Crewe and J. Wall, "A scanning microscope with 5 Å resolution," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 375–393, 1970.

[48] P. D. Nellist and S. J. Pennycook, "Incoherent imaging using dynamically scattered coherent electrons," *Ultramicroscopy*, vol. 78, no. 1-4, pp. 111–124, 1999.

[49] M. Isaacson and D. Johnson, "The microanalysis of light elements using transmitted energy loss electrons," *Ultramicroscopy*, vol. 1, no. 1, pp. 33–52, 1975.

[50] A. J. D. Alfonso, B. Freitag, D. Klenov, L. J. Allen, A. J. D'Alfonso, B. Freitag, D. Klenov, and L. J. Allen, "Atomic-resolution chemical mapping using energy-dispersive x-ray spectroscopy," *Physical Review B - Condensed Matter and Materials Physics*, vol. 81, no. 10, pp. 2–5, 2010.

[51] S. J. Pennycook and L. A. Boatner, "Chemically sensitive structure-imaging with a scanning transmission electron microscope," *Nature*, vol. 336, no. 6199, pp. 565–567, 1988.

[52] A. Howie, "Image contrast and localized signal selection techniques," *Journal of Microscopy*, vol. 117, no. September, pp. 11–23, 1979.

[53] S. J. Pennycook and D. E. Jesson, "High-resolution Z-contrast imaging of crystals," *Ultramicroscopy*, vol. 37, no. 1-4, pp. 14–38, 1991.

[54] S. S. J. Pennycook, D. E. Jesson, M. F. Chisholm, N. D. Browning, A. J. Mcgibbon, and M. M. McGibbon, "Z-contrast imaging in the scanning transmission electron microscope," *Microscopy and Microanalysis*, vol. 1, no. 6, pp. 231–251, 1995.

[55] E. J. Kirkland, R. F. Loane, and J. Silox, "Simulation of Annular Dark Fiels STEM Images using a Modified Multislice Method," *Ultramicroscopy*, vol. 23, no. 1, pp. 77–93, 1987.

[56] H. J. Lipkin, "Phase uncertainty and loss of interference in a simple model for mesoscopic Aharonov-Bohm experiments," *Physical Review A*, vol. 42, no. 1, pp. 49–54, 1990.

[57] J. M. Lebeau, S. D. Findlay, X. Wang, A. J. Jacobson, L. J. Allen, and S. Stemmer, "High-angle scattering of fast electrons from crystals containing heavy elements:

Simulation and experiment," *Physical Review B - Condensed Matter and Materials Physics*, vol. 79, no. 21, pp. 2–7, 2009.

[58] A. B. Yankovich, B. Berkels, W. Dahmen, P. Binev, S. I. Sanchez, S. A. Bradley, A. Li, I. Szlufarska, and P. M. Voyles, "Picometre-precision analysis of scanning transmission electron microscopy images of platinum nanocatalysts," *Nature Communications*, vol. 5, no. May, pp. 1–7, 2014.

[59] P. D. Nellist, "Scanning Transmission Electron Microscopy," in *Scanning Transmission Electron Microscopy* (S. J. Pennycook and P. D. Nellist, eds.), no. i, pp. 91–116, Springer Science+Business Media, LLC, 2011.

[60] S. Van Aert, K. J. Batenburg, M. D. Rossell, R. Erni, and G. Van Tendeloo, "Three-dimensional atomic imaging of crystalline nanoparticles," *Nature*, vol. 470, no. 7334, pp. 374–377, 2011.

[61] R. Hegerl and W. Hoppe, "Influence of electron noise on three-dimensional image reconstruction," *Zeitschrift für Naturforschung Teil A*, vol. 31, no. 3, pp. 1717–1721, 1976.

[62] L. Jones and P. D. Nellist, "Identifying and correcting scan noise and drift in the scanning transmission electron microscope," *Microscopy and Microanalysis*, vol. 19, no. 4, pp. 1050–1060, 2013.

[63] L. Jones, H. Yang, T. J. Pennycook, M. S. J. Marshall, S. Van Aert, N. D. Browning, M. R. Castell, and P. D. Nellist, "Smart Align—a new tool for robust non-rigid registration of scanning microscope data," *Advanced Structural and Chemical Imaging*, vol. 1, no. 1, p. 8, 2015.

[64] J. Sprinzak, "ScienceDirect.com - Pattern Recognition Letters - Affine point matching," *Pattern Recognition Letters*, vol. 15, no. April, pp. 337–339, 1994.

[65] A. Rečnik, G. Möbus, and S. Šturm, "IMAGE-WARP: A real-space restoration method for high-resolution STEM images using quantitative HRTEM analysis," *Ultramicroscopy*, vol. 103, no. 4, pp. 285–301, 2005.

[66] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*, vol. 6256. Springer, 7th ed ed., 2009.

[67] P. A. Midgley and M. Weyland, "3D electron microscopy in the physical sciences: The development of Z-contrast and EFTEM tomography," in *Ultramicroscopy*, vol. 96, pp. 413–431, 2003.

[68] J. Radon, "Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten [On the Determination of Functions From Their Integral Values Along Certain Manifolds]," *Berichte der Sachsischen Akadamie der Wissenschaft*, vol. 69, pp. S262–S277, 1917.

[69] G. T. Herman, *Image reconstruction from projections. Implementation and applications.*, vol. 1. Springer, 1979.

[70] P. Gilbert, "Iterative methods for the three-dimensional reconstruction of an object from projections," *Journal of Theoretical Biology*, vol. 36, no. 1, pp. 105–117, 1972.

[71] R. Gordon, R. Bender, and G. T. Herman, "Algebraic Reconstruction Techniques (ART) for three-dimensional electron microscopy and X-ray photography," *Journal of Theoretical Biology*, vol. 29, no. 3, pp. 471–481, 1970.

[72] A. H. Andersen and A. C. Kak, "SIMULTANEOUS ALGEBRAIC RECONSTRUCTION TECHNIQUE (SART): a Superior Implementation of the Art Algorithm," *Ultrasonic Imaging*, vol. 94, pp. 81–94, 1984.

[73] W. Hoppe, H. J. Schramm, M. Sturm, N. Hunsmann, and J. Gaßmann, "Three-Dimensional Electron Microscopy of Individual Biological Objects Part I. Methods," *Zeitschrift fur Naturforschung - Section A Journal of Physical Sciences*, vol. 31, no. 6, pp. 645–655, 1976.

[74] W. Van den Broek, A. Rosenauer, B. Goris, G. T. Martinez, S. Bals, S. Van Aert, and D. Van Dyck, "Correction of non-linear thickness effects in HAADF STEM electron tomography," *Ultramicroscopy*, vol. 116, pp. 8–12, 2012.

[75] J. Miao, F. Förster, and O. Levi, "Equally sloped tomography with oversampling reconstruction," *Physical Review B - Condensed Matter and Materials Physics*, vol. 72, no. 5, pp. 3–6, 2005.

[76] A. Pryor, Y. Yang, A. Rana, M. Gallagher-Jones, J. Zhou, Y. H. Lo, G. Melinte, W. Chiu, J. A. Rodriguez, and J. Miao, "GENFIRE: A generalized Fourier iterative reconstruction algorithm for high-resolution 3D imaging," *Scientific Reports*, vol. 7, no. 1, p. 10409, 2017.

[77] R. Leary, Z. Saghi, P. A. Midgley, and D. J. Holland, "Compressed sensing electron tomography," *Ultramicroscopy*, vol. 131, pp. 70–91, 2013.

[78] J. R. Jinschek, K. J. Batenburg, H. A. Calderon, R. Kilaas, V. Radmilovic, and C. Kisielowski, "3-D reconstruction of the atomic positions in a simulated gold nanocrystal based on discrete tomography: Prospects of atomic resolution electron tomography," *Ultramicroscopy*, vol. 108, no. 6, pp. 589–604, 2008.

[79] G. T. Herman and A. Kuba, *Discrete Tomography—Foundations, Algorithms, and Applications*, vol. 53. Birkhäuser, Boston, 1999.

[80] G. T. Herman and A. Kuba, *Advances in Discrete Tomography and its Applications*. Birkhäuser, Boston, 2007.

[81] S. D. Findlay, S. Azuma, N. Shibata, E. Okunishi, and Y. Ikuhara, "Direct oxygen imaging within a ceramic interface, with some observations upon the dark contrast at the grain boundary," *Ultramicroscopy*, vol. 111, no. 4, pp. 285–289, 2011.

[82] J. Frank, *Three-dimensional electron microscopy of macromolecular assemblies: visualization of biological molecules in their native state.* Oxford University Press, 2006.

[83] J. Frank, *Electron tomography - Three-Dimensional Imaging with the Transmission Electron Microscope.* Springer Science+Business Media, LLC, 1992.

[84] C. C. B. Murray W. Mahoney, "Fundamentals of Diffusion Bonding," in *ASM Handbook Vol.6 Welding, Brazing, and Soldering*, pp. 156–159, ASM International, 1993.

[85] K. L. Merkle and D. J. Smith, "Atomic structure of symmetric tilt grain boundaries in NiO," *Physical Review Letters*, vol. 59, no. 25, pp. 2887–2890, 1987.

[86] N. Shibata, T. Yamamoto, Y. Ikuhara, and T. Sakuma, "Structure of [110] Tilt Grain Boundaries in Zirconia Bicrystals Full-length paper Structure of [110] tilt grain boundaries in zirconia bicrystals," *Journal of Electron Microscopy*, vol. 50, no. 6, pp. 429–433, 2001.

[87] N. Shibata, F. Oba, T. Yamamoto, Y. Ikuhara, and T. Sakuma, "Atomic structure and solute segregation of a $\Sigma = 3$, [110]/{111} grain boundary in an yttria-stabilized cubic zirconia bicrystal," *Philosophical Magazine Letters*, vol. 82, no. 7, pp. 393–400, 2002.

[88] B. Feng, I. Sugiyama, H. Hojo, H. Ohta, N. Shibata, and Y. Ikuhara, "Atomic structures and oxygen dynamics of CeO2 grain boundaries," *Scientific Reports*, vol. 6, no. December 2015, pp. 1–7, 2016.

[89] R. Ishikawa, A. R. Lupini, S. D. Findlay, and S. J. Pennycook, "Quantitative annular dark field electron microscopy using single electron signals.," *Microscopy and microanalysis : the official journal of Microscopy Society of America, Microbeam Analysis Society, Microscopical Society of Canada*, vol. 20, no. 1, pp. 99–110, 2014.

[90] S. Paciornik, R. Kilaas, J. Turner, and U. Dahmen, "A pattern recognition technique for the analysis of grain boundary structure by HREM," *Ultramicroscopy*, vol. 62, no. 1-2, pp. 15–27, 1996.

[91] A. J. Den Dekker, S. Van Aert, A. Van Den Bos, and D. Van Dyck, "Maximum likelihood estimation of structure parameters from high resolution electron microscopy images. Part I: A theoretical framework," *Ultramicroscopy*, vol. 104, no. 2, pp. 83–106, 2005.

[92] S. Bals, S. Van Aert, G. Van Tendeloo, D. Ávila-Brande, S. V. Aert, G. V. Tendeloo, A. David, S. Van Aert, G. Van Tendeloo, and D. Ávila-Brande, "Statistical estimation of atomic positions from exit wave reconstruction with a precision in the picometer range," *Physical Review Letters*, vol. 96, no. 9, pp. 1–4, 2006.

[93] A. McGibbon, S. J. Pennycook, and D. Jesson, "Crystal structure retrieval by maximum entropy analysis of atomic resolution incoherent images," *Journal of Microscopy*, vol. 195, no. July, pp. 44–57, 1999.

[94] S. D. Findlay, N. Shibata, H. Sawada, E. Okunishi, Y. Kondo, and Y. Ikuhara, "Dynamics of annular bright field imaging in scanning transmission electron microscopy," *Ultramicroscopy*, vol. 110, no. 7, pp. 903–923, 2010.

[95] K. Kimoto, T. Asaka, X. Yu, T. Nagai, Y. Matsui, and K. Ishizuka, "Local crystal structure analysis with several picometer precision using scanning transmission electron microscopy," *Ultramicroscopy*, vol. 110, no. 7, pp. 778–782, 2010.

[96] Y. M. Kim, J. He, M. D. Biegalski, H. Ambaye, V. Lauter, H. M. Christen, S. T. Pantelides, S. J. Pennycook, S. V. Kalinin, and A. Y. Borisevich, "Probing oxygen vacancy concentration and homogeneity in solid-oxide fuel-cell cathode materials on the subunit-cell level," *Nature Materials*, vol. 11, no. 10, pp. 888–894, 2012.

[97] B. Berkels, P. Binev, D. A. Blom, W. Dahmen, R. C. Sharpley, and T. Vogt, "Optimized imaging using non-rigid registration," *Ultramicroscopy*, vol. 138, pp. 46–56, 2014.

[98] B. Berkels and B. Wirth, "Joint denoising and distortion correction of atomic scale scanning transmission electron microscopy images," *CoRR*, pp. 1–35, 2016.

[99] D. J. Hand, G. J. McLachlan, and K. E. Basford, "Mixture Models: Inference and Applications to Clustering.," *Applied Statistics*, vol. 38, no. 2, p. 384, 1989.

[100] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2012.

[101] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*. John Wiley & Sons, Ltd, 2011.

[102] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 2014.

[103] I. Markovsky and S. Van Huffel, "Overview of total least-squares methods," *Signal Processing*, vol. 87, no. 10, pp. 2283–2302, 2007.