Department of Construction Sciences
Solid Mechanics

ISRN LUTFD2/TFHF-21/6001-SE(1-40)

Exploring intra-granular strain in granular systems under mechanical load using scanning 3DXRD

Bachelor's Dissertation by

Philip Vestin

Supervisors:
Dr. Stephen Hall, Division of Solid Mechanics, Faculty of Engineering, LTH
M.Sc. Axel Henningsson, Division of Solid Mechanics, Faculty of Engineering, LTH

Examiner:
Prof. Matti Ristinmaa, Division of Solid Mechanics, Faculty of Engineering, LTH

Printed by Media-Tryck AB, Lund, Sweden
For information, address:
Division of Solid Mechanics, Lund University, Box 118, SE-221 00 Lund, Sweden
Webpage: www.solid.lth.se

Dissertation for the degree of Bachelor of Science in Engineering

Exploring intra-granular strain in granular systems under mechanical load using scanning 3DXRD

Philip Vestin
ph7705ve-s@student.lu.se
June 28, 2021

Keywords

Granular mechanics, 3DXRD, x-ray diffraction, Python

Abstract

This dissertation serves to explain the theory behind three-dimensional scanning x-ray diffraction while at the same time connecting the theory to a software package used for processing data obtained from measurements using scanning x-ray diffraction. This software package is then used and built upon to analyse the data obtained from the measurement of a sample of 12 silica grains subject to an uniaxial load of 60 N. The results reveal the internal strain field heterogeneity in the grains. From these strain fields it will be possible to identify grains that are at risk of fracturing, examine the interaction of grains at the surfaces of intra-granular contacts and in general examine the strain both at the surface of the grains and inside the grains themselves, demonstrating the usefulness of scanning x-ray diffraction to study the fine details of granular mechanics.

Acknowledgements

The author would like to express his gratitude towards his supervisors Dr. S.Hall and M.Sc. A.Henningson for the opportunity to take part of their research regarding three-dimensional x-ray diffraction and to write this Bachelor's dissertation. The time they have taken to answer the many questions that arose during this project is much appreciated.

Contents

Chapter 1

Introduction

As described by Poulsen [1], three dimensional x-ray diffraction (3DXRD) is a method of measuring properties of crystalline materials. By illuminating the sample that is studied with a high-intensity x-ray beam and studying the diffracted x-rays from the sample, conclusions can then be made about the sample composition and other properties. When the beam size is decreased to a size smaller than the size of the individual grains of the specimen, properties can be observed on an intra-granular scale.

Coupled with a strain reconstruction technique, algebraic strain refinement (ASR), as developed by Henningson *et al*[2], the internal strains of the sample can be measured. Internal strain reconstruction using ASR has previously been tested for a horizontal section (slice) of the grain assembly. The objective for this project is to process the data for the entire volume of the grain assembly.

The following work is a bachelors dissertation in engineering, with focus on solid mechanics and x-ray physics, striving to explain, in a concise way, the method of 3DXRD together with obtained experimental results. This is done by first giving the necessary prerequisites within the fields of crystallography and tomography, followed by an explanation of the necessary theory for 3DXRD while simultaneously connecting the theory to a software package intended for analysis of data obtained from 3DXRD, implemented using the Python programming language. Finally the author's own contributions to the software will be explained and relevant results will be presented.

Due to the large amount of preparatory data processing that needs to be done and the fact that large software packages rarely are built by a single contributor, certain aspects of the 3DXRD software relating to preprocessing of data are omitted. Also, any technical information regarding how to practically perform measurements using synchrotron x-ray radiation, including but not limited to calibration, operating procedures and necessary equipment, will also be left out. However, a basic description of the geometry will be necessary and can be found in chapter 3.

The experiment considered in the analysis consisted of 12 silica grains inside a load chamber, which allowed an adjustable force to be applied to the collection of grains by pushing downwards on the top of the grain assembly. The grains were analyzed using 3DXRD measurments performed at the ESRF ID11 beamline.

Measurements were conducted by a team led by Stephen Hall and the software necessary for analyzing the data were provided in the s3dxrd package by Axel Henningsson[3] and in the ImageD11 package by Jon Wright [4]. Data were available for several different applied loads, but due to time constraints only the data from when the sample was subject to a load of 60 N has been analysed within the frame of the project on which this dissertation is based.

At the start of the project, the 3DXRD analysis had only been tested on a single slice of the dataset. The challenge in this project was to verify the functionality of the 3DXRD approach and software as well as to apply it to the whole 3D volume analysis, assembling all slices into a three-dimensional structure. Consequently the 3D strains could be visualized over a complete granular assembly under load for the first time. Strains are thus visualized across slices and strain behaviour at grain surfaces or contact points can be investigated.

Chapter 2

Crystals and their internal structure

2.1 The crystal lattice and basis

A common structure of materials is one where the atoms are regularly ordered in a repeating pattern. By defining a stencil of atoms arranged in a predefined configuration and repeating this stencil, an arbitrarily large piece of the material can be created. This is known as a material exhibiting a crystal structure.

In order to define a crystal, one must first define the lattice. The lattice is a purely conceptual construction that consists of lattice sites with a specified distance between them. As described by Borchardt-Ott [5] the one-dimesional lattice consists of a line with evenly spaced lattice sites. Traversing the lattice is done by moving a distance a from one lattice site to the next. The stencil that is repeated in order to construct the one-dimesional lattice is to place a lattice site on a line with a distance of a to the next lattice site. This stencil is formally called the unit cell and can be described by the lattice parameters, which in the one-dimensional case consists of the single lattice parameter a. As will be seen, the concept of the unit cell can be generalized to two and three dimensions.

In two dimensions, a lattice can be created by repeating the previous one-dimensional lattice in the transverse direction. Note that the lattice points do not necessarily need to be placed such that vectors originating from a common lattice site, but with different directions, are either parallel or orthogonal. Nor does the spacing between the lattice sites need to be the same in both directions. Hence, the two dimensional lattice needs three lattice parameters in order to be described.

Keeping to the method for describing the crystal lattice explained by Borchardt-Ott[5], an arbitrary lattice site in the two-dimensional lattice is chosen as the origin. Defining the vectors **a** and **b**, directed from the origin to the nearest lattice site in nonparallel directions, we get the two lattice parameters a and b, consisting of the length of the corresponding vectors. The third lattice parameter, γ is the angle between the vectors **a** and **b**.

In three dimensions, the lattice is generated by stacking the two-dimensional lattice in a direction perpendicular to the plane that the two-dimensional lattice exists in. From the origin of the lattice, another vector can be traced to the nearest lattice site that does not lie in the plane defined by the vectors **a** and **b**. The third vector is commonly denoted by **c** and is chosen in the direction that causes the vectors **a**, **b** and **c** to form a right-handed coordinate system. Expansion of the two-dimensional lattice to three dimensions introduces in total three new lattice parameters. These consist of c, the length of the new vector **c**, β, the angle between the vectors **a** and **c**, and finally α, the angle between the vectors **b** and **c**. These three lattice parameters together with the previously defined lattice parameters a, b and γ uniquely define the three-dimensional unit cell.

One may ask which lattice sites that belongs to which unit cell. This depends largely on how the unit cell is chosen. Described above is the primitive unit cell where only one lattice site belongs to each unit cell. Nielsen and McMorrow [6] demonstrates this by imagining that the unit cell is translated a small amount in an arbitrary direction. Such a translation would only cause one new lattice site to be within the volume of the unit cell, no matter how the unit cell is translated. Hence, each lattice site is associated with only one unit cell.

The crystal lattice merely serves as a concept to describe the structure of the material. In order to actually construct a crystal, atoms needs to be assigned to selected locations in the lattice. As described by Borchardt-Ott, the collection of atoms associated with a unit cell is referred to as a basis and can be described in the coordinate system given by the vectors **a**, **b** and **c** [5]. This basis is repeated in every unit cell, thus the entire crystal can be constructed once the lattice and the basis have been defined.

2.2 Crystallographic planes and Miller indices

The three-dimensional lattice allows for several different planes in the lattice to be identified. Commonly, this is done by letting the vectors **a**, **b** and **c** act as the crystallographic axes. As defined by Nielsen and McMorrow[1][6], each lattice site can be defined using the crystallographic axes and the chosen origin.

[1]Since the notation regarding the crystallographic lattice varies across the literature, the author decides to stay with the notation used by Borchardt-Ott, and later by Poulsen, rather than introducing a new, but equivalent notation used by Nielsen and McMorrow. Any reader that looks up Nielsen and McMorrow will find that the notation is slightly different.

An arbitrary lattice site can then be described using vector notation as

$$\mathbf{R} = n_1\mathbf{a} + n_2\mathbf{b} + n_3\mathbf{c}.$$

The crystal planes are labeled by examining the points of intersection between the planes and the crystallographic axes. Commonly, the reciprocals of the coordinates (n_1, n_2, n_3) are used to label the crystal plane. As described by Borchardt-Ott [5] these reciprocal coordinates are named Miller indices and are labeled as (h, k, l), where $h = 1/n_1$, $k = 1/n_2$ and $l = 1/n_3$. It is worth noting that the Miller indices are chosen such that they are the lowest possible integer values that can describe a given crystal plane. For example, $(\frac{1}{2}, 1, 1)$ would be written as $(1, 2, 2)$. Intercepts with a negative coordinate as measured in the coordinate system defined by the crystallographic axes are denoted by a bar over the corresponding Miller index, e.g $(-1, 1, 1)$ is written as $(\bar{1}, 1, 1)$.

2.3 The reciprocal lattice

Since the Miller indices are defined as the reciprocals of the coordinates in real space, one may construct a lattice in reciprocal space, i.e a fictitious spatial dimension where the coordinates of a point are described using the Miller indices. In order to construct such a spatial dimension, basis vectors needs to be defined. Nielsen and McMorrow defines these reciprocal basis vectors as[6]

$$\mathbf{a}^* = 2\pi\frac{\mathbf{b}\times\mathbf{c}}{\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})},\ \ \mathbf{b}^* = 2\pi\frac{\mathbf{c}\times\mathbf{a}}{\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})},\ \ \mathbf{c}^* = 2\pi\frac{\mathbf{a}\times\mathbf{b}}{\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})}.$$

With the reciprocal lattice basis vectors defined, one can define an arbitrary point in the reciprocal lattice much in the same way as in the real space lattice. The reciprocal lattice vectors can then be described as

$$\mathbf{G} = h\mathbf{a}^* + k\mathbf{b}^* + l\mathbf{c}^*.$$

The factor of 2π gives the reciprocal lattice vectors the property that when scalar multiplication between a reciprocal lattice vector and a real space lattice vector is performed, the result will be an integer multiple of 2π[6]. As will be seen, this gives the reciprocal lattice vectors some much desired properties when calculating the possible ways an incoming ray of light may diffract in the crystal planes. Just like the real-space lattice, one may define a reciprocal lattice unit cell and corresponding reciprocal lattice constants.

It turns out to be useful to gather these reciprocal lattice parameters in a matrix, commonly denoted by $\mathbf{B}$, in such a way that the $\mathbf{B}$ matrix acts as a transformation matrix, transforming a vector of Miller indices to a scattering vector given in a Cartesian coordinate system defined for the grain that scattered the x-ray beam. The $\mathbf{B}$ matrix is defined as follows[7],

$$\mathbf{B} = \begin{bmatrix} a^* & b^*\cos(\gamma^*) & c^*\cos(\beta^*) \\ 0 & b^*\sin(\gamma^*) & -c^*\sin(\beta^*)\cos(\alpha) \\ 0 & 0 & c^*\sin(\beta^*)\sin(\alpha) \end{bmatrix} \quad \text{where} \quad \cos(\alpha) = \frac{\cos(\beta^*)\cos(\gamma^*) - \cos(\alpha^*)}{\sin(\alpha^*)\sin(\beta^*)}.$$

2.4 Diffraction and scattering in crystalline materials

When x-rays are incident on a single atom, elastically scattered x-rays may be radiated from the atom due to the electromagnetic forces acting on the atomic electrons, caused by the incident x-rays [6]. Here, elastic means that the scattered x-ray beam will have the same energy as the incident x-ray beam.

As described by Nielsen and McMorrow [6], a convenient way of describing an incident electromagnetic wave is by its wavevector $\mathbf{k}$. Similarily, the wavevector of the scattered wave may be denoted by $\mathbf{k}'$. The difference between $\mathbf{k}'$ and $\mathbf{k}$ gives a wavevector transfer vector, or scattering vector, here denoted by $\mathbf{Q}$.

By letting the vector directed from the origin of the atom to the location of the scattering event be denoted by $\mathbf{r}$, the phase difference between radiation scattered at the origin and at the location designated by $\mathbf{r}$ can be calculated as the scalar product between $\mathbf{Q}$ and $\mathbf{r}$[6]. Taking a quantum mechanical approach to the situation, letting the atomic electrons be represented as a charge distribution, the atomic form factor, $f^0(\mathbf{Q})$, of an atom with a charge density of $\rho(\mathbf{r})$ can be calculated by integrating over the entire volume of the atom as[6]

$$f^0(\mathbf{Q}) = \int \rho(\mathbf{r}) e^{i\mathbf{Q}\cdot\mathbf{r}} d\mathbf{r}.$$

The concept of atomic scattering can be generalised to an entire crystal due to the symmetric nature of the crystal structure. Scattering occurs from the individual atoms of the crystal, but as previously discussed the crystal consists of the same unit cell of atoms arranged repeatedly in a lattice. Hence the scattering will be repeated and amplified for each lattice site. By separating the resulting scattering amplitude into a sum over all lattice sites and a sum over the internal structure of the unit cell, as done by Nielsen and McMorrow [6], the scattering amplitude for a crystal structure can be expressed as

$$F(\mathbf{Q}) = \sum_j f_j(\mathbf{Q}) e^{i\mathbf{Q}\cdot\mathbf{r}_j} \sum_n e^{i\mathbf{Q}\cdot\mathbf{R}_n}. \tag{2.1}$$

The second term in equation (2.1) will be decisive for the scattering amplitude. While the first term only depends on the location of the atoms inside the unit cell and the corresponding atomic form factors of said atoms, the second term depends on the scattering vector in connection with the lattice vectors. If $\mathbf{Q}\cdot\mathbf{R}_n$ varies randomly as complex numbers on the unit circle, the sum will, on average, be of order unity[6]. In order for $F(\mathbf{Q})$ to adopt significantly larger values, the sum over the lattice sites needs to add up coherently. This happens when $\mathbf{Q}\cdot\mathbf{R}_n = 2\pi m$, where m is an integer. This, however, is exactly the quality of the reciprocal lattice vectors. Therefore, the scattering amplitude for a crystalline material increases when the scattering vector happens to coincide with a reciprocal lattice vector, i.e $\mathbf{Q} = \mathbf{G}$. This is commonly referred to as the Laue condition or Laue equation and, as done by Nielsen and McMorrow[6], can be shown to be a vector valued counterpart to Bragg's law, $m\lambda = 2d\cdot\sin(\theta)$, where m is an integer, λ is the wavelength of the incident light and d is the spacing between the atomic planes in the crystal.

Chapter 3

Three-dimensional scanning x-ray diffraction

3.1 Introduction

Three-dimensional x-ray diffraction, similar to the method described by Poulsen [1], together with a strain reconstruction algorithm, ASR described by Henningson *et al* [2], are the foundation for constructing a three-dimensional reconstruction of the intra-granular strain fields. Here, the theory behind the scanning 3DXRD-method will be explained in the context of the properties regarding crystalline materials discussed in chapter 2. The ASR algorithm can then be explained, starting from the quantities gathered from the 3DXRD measurements. To help explain the structure of the data processing and the s3dxrd-package, a flowchart has been attached as appendix A.

3.2 3DXRD-algorithm and experimental setup

The geometry of the experimental setup used for the considered measurements can be seen in Figure 3.1. An x-ray beam with a square cross-section and a side length of 25 microns illuminates a parallelepiped of the sample. Any parts of a grain whose alignment means that there will be a diffraction plane oriented in such a way that the Laue condition will be fulfilled is going to produce a diffraction spot on a digital detector screen placed behind the sample. The diffraction spots are defined by giving a threshold for what intensity is necessary for a detector pixel to be classified as being illuminated, and then searching for connected pixel areas with an intensity higher than the intensity threshold. Since the diffraction spots consist of two-dimensional areas in the detector plane, a center-of-mass can be calculated for the diffraction spot[1]. This makes the data significantly more sparse, as only a single point on the detector is stored for each diffraction spot together with the sum of the recorded intensity, and the position of the sample.

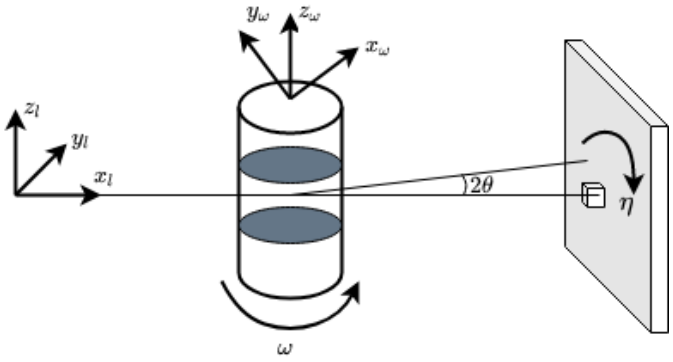


Figure 3.1: Geometry of the 3DXRD experimental setup. Coordinates with the subscript l represent the laboratory coordinate system, coordinates with the subscript ω represent a coordinate system rotating with the sample. The incoming x-ray is directed along the x_l-axis and the other coordinate axes of the laboratory system are chosen such that a right-handed coordinate system is created, with the z-axis directed upwards and parallel to the axis of rotation of the sample.

Since the beam used has a square cross-section, the positive y-values, as seen in Figure 3.1 are all scanned for a given ω-setting. For each scanned y-position, the sample is rotated $d\omega$ degrees around the current ω setting. Hence, what is observed for a single scanned y-position is the integrated intensity obtained in an angular interval of $d\omega$. For the data used in this dissertation, the angular step $d\omega$ was set to 0.05 degrees.

Once the entire range of positive y-values has been scanned, the sample ω-position is advanced by an increment of $\cdot d\omega$. In total, the sample is rotated in an ω interval ranging from $-180°$ to $180°$. Once the scanning procedure has been repeated for all ω settings, the sample is translated in the z-direction, as defined from Figure 3.1 and the sample is scanned at the new z-position. This procedure is repeated until the entire sample has been scanned

Just like when mapping scattering vectors between the grain and the sample coordinate system, as described in chapter 2, the scattering vectors can be mapped to the laboratory system and the ω coordinate system from the sample coordinate system. The transformation matrix between the ω coordinate system and the laboratory system is given from the geometrical conditions of the experimental setup. Since the geometry is identical to the one described by Henningson *et al*[2] the transformation matrix is given as

$$\mathbf{\Omega} = \begin{bmatrix} \cos(\omega) & -\sin(\omega) & 0 \\ \sin(\omega) & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

which operates on the measured scattering vectors in the ω coordinate system according to

$$\mathbf{G}_l = \mathbf{\Omega}\mathbf{G}_\omega. \tag{3.1}$$

Another transformation matrix that is of importance is the **U** matrix. This matrix represents the orientation of the grain and maps a scattering vector measured in the Cartesian coordinate system of the grain to another Cartesian coordinate system that is defined for the entire sample[7]. The orientation matrix **U** is not a known quantity. Instead it must be calculated and refined as part of the indexing algorithm. Regarding the coordinate transforms, the following relationships apply, where $\mathbf{G}_{hkl}$ is defined in equation 2.3,

$$\mathbf{G}_c = \mathbf{B}\mathbf{G}_{hkl} \text{ and } \mathbf{G}_s = \mathbf{U}\mathbf{G}_c. \tag{3.2}$$

In the descriptions of the coordinate transformations as given by Poulsen *et al*[8] and Lauridsen *et al* [7], a matrix labeled **S** is used to transform scattering vectors measured in the sample coordinate system to the ω coordinate system according to $\mathbf{G}_\omega = \mathbf{S}\mathbf{G}_s$. However, due to the laboratory setup being the same as for Henningson *et al*[2], the sample system and the ω coordinate system coincide at all times. Hence the matrix **S** may be considered to be equivalent with the identity matrix. This allows for equation (3.2) to be summarised as

$$\mathbf{G}_\omega = \mathbf{U}\mathbf{B}\mathbf{G}_{hkl}. \tag{3.3}$$

In total, the transformation of scattering vectors measured as Miller indices in reciprocal space, to scattering vectors measured in the real-space laboratory coordinate system is found by combining equations (3.3) and (3.1) to yield

$$\mathbf{G}_l = \mathbf{\Omega}\mathbf{U}\mathbf{B}\mathbf{G}_{hkl}.$$

The recorded diffraction spots give information about the strain state of the sample. As shown by Poulsen [1] and later used by Henningson, *et al*[2] the average strain in a direction perpendicular to the diffracting planes, sampled from the area illuminated by the x-ray beam, can be estimated to good precision from the measured Bragg angles as

$$\bar{\varepsilon} = \frac{sin(\theta_r) - sin(\theta)}{sin(\theta)}, \tag{3.4}$$

where θ_r is a reference Bragg angle for a strain-free reference state.

3.3 Tomographic reconstruction principles

When the sample is illuminated by the x-ray beam, grains that are oriented in such a way that crystal planes fulfill the Laue condition will cause a diffracted ray to impinge on the detector screen. The recorded reflections contain not only information about the orientations of the grains, but also information about the shape of the grains. As will be seen in chapter 4, the strain state of the illuminated region can also be extracted from the recorded diffraction spots and the tomographic reconstructions. Recreating an image from its projection is called tomography and the following section will provide the necessary prerequisites for understanding the measurements and data processing to reconstruct the grain topology. For a more in-depth treatment of tomography see, for example, Kak and Slaney[9].

Tomographic reconstruction is based on the fact that the 2D image of a diffracted x-ray beam on a detector screen may be considered as the result of a line integral. A single measurement only contains information about how the material responds to x-ray illumination in the volume traced out by the x-ray beam. If a slice of the material is to be reconstructed, the first step in obtaining the reconstruction is to repeat the measurement along the length of the sample. This corresponds to moving the sample along the y_l axis, see Figure 3.2.

To collapse the two-dimensional diffraction spots into a single line, where each point on the line corresponds to the value of the aforementioned line integral, the intensity of the diffraction spots is summed and any deviation in the z_l-direction, as defined in Figure 3.2, is discarded. Hence, the diffraction spots, which previously were spread out over the entire detector plane, are projected onto a line. To obtain a data set from which information at a voxel level (where a voxel is a cubic region in 3D space) can be extracted, the sample is rotated so that it is imaged in an angular range from $-180°$ to $180°$.

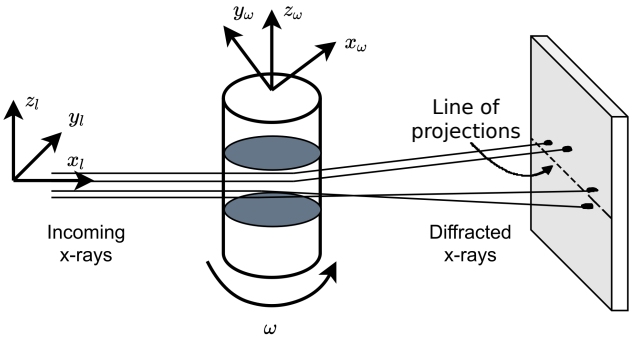


Figure 3.2: Illustration of illumination of a slice using a sequence of x-ray beams. The diffracted x-rays are spread out in the detector plane, but will all be projected down on a single line parallel to the y-axis.

For each angular value the sample is scanned along the positive y_l axis. As done by Poulsen and Schmidt [10], the lines resulting from scanning the sample throughout the entire angular range can be stacked in a chart, where one axis represents the angle of the of the sample and the other axis represents the coordinate along the length of the sample. These charts are commonly referred to as sinograms, due to the shapes of the resulting intensity distributions. An example of a sinogram and the reconstruction obtained from that sinogram can be seen in Figure 3.3.

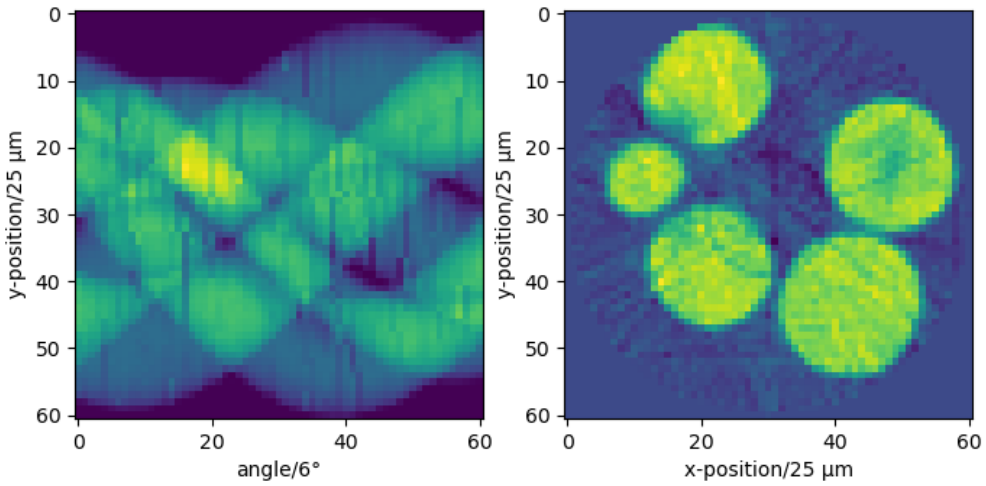


Figure 3.3: To the left: An example of a sinogram. To the right: The reconstruction resulting from the sinogram shown to the left. The angles are recorded in the interval -180°to 180°, but has been binned into a smaller interval.

The sinograms may be considered the result of applying the Radon transform to the original shape. The Radon transform is related to the two-dimensional Fourier transform and may be inverted, such that applying the inverse Radon transform reproduces the original image. The procedure of measuring the projections at different angles and applying the inverse Radon transform is known as the filtered back-projection algorithm[9]. Since the properties of the Fourier transform are well examined, and efficient computer implementations exist in the form of fast Fourier transform algorithms (FFT-algorithms), performing these operations is possible on an ordinary computer[11].

3.4 Preprocessing

Since no three-dimensional mapping of diffraction spots to corresponding grain positions in the sample has been made yet, all diffraction spots for a given z-slice are assumed to have originated from the center point of the sample as far as positioning in the xy-plane is concerned. The diffracted rays can then be traced from their centers of mass at the detector screen, to the center of the slice that the diffraction spots were recorded for. This tracing procedure together with the known geometry of the experimental setup gives the approximate Bragg angle and scattering vectors. From the position of the diffraction spots on the detector, the Miller indices of the different diffraction spots can be deduced based on previously established knowledge of the crystallographic properties of the material.

The method for indexing the diffraction spots is described by Lauridsen *et al*[7] and is based on equation (3.3). It is noted that the reciprocal lattice parameters are given quantities when measurements are to be performed, otherwise the grain indexing algorithm will not have sufficient information about the material that is examined. Since Lauridsen *et al* use the **S** matrix, the formulation given as equation 9 by Lauridsen *et al* is equivalent to equation (3.3), as given here.

The indexing procedure starts by determining the grain orientations. This is done by sampling all possible orientation matrices **U** and assigning scattering vectors to the grain orientation matrix by counting the number of $\mathbf{G}_{hkl}$-vectors that fulfill equation (3.3) for at least one $\mathbf{G}_{\omega}$-vector. As explained by Lauridsen *et al*[7], two different criteria are used to identify which orientation matrices correspond to grains in the sample. The first criteria being a completeness critera, stating that $M_{exp} \geq (1-\alpha)M_0$, where M_{exp} is the counted number of $\mathbf{G}_{hkl}$ scattering vectors, M_0 is the theoretically possible number of reflections and α is a coefficient of tolerance. The second criteria is a uniqueness criteria, stating that a set of matching $\mathbf{G}_{hkl}$-vectors for a given **U** can not be a subset of another set of $\mathbf{G}_{hkl}$-vectors found for a different grain orientation matrix.

The next step is to sort the $\mathbf{G}_{hkl}$-vectors found for the different **U**-matrices according to how many observed $\mathbf{G}_{\omega}$-vectors that the $\mathbf{G}_{hkl}$-vectors might correspond to. The ambiguity arises since there is a need for tolerances when measuring the diffraction spots and that the number of sampled **U**-matrices is finite, in contrast to the infinite number of possible grain orientations. Three different categories are used to group the different $\mathbf{G}_{hkl}$-vectors. Group A contains the $\mathbf{G}_{hkl}$-vectors that do not correspond to any $\mathbf{G}_{\omega}$-vectors. Group B contains the $\mathbf{G}_{hkl}$-vectors that correspond to exactly one matching $\mathbf{G}_{\omega}$-vector and group C contains the $\mathbf{G}_{hkl}$-vectors that correspond to two or more $\mathbf{G}_{\omega}$-vectors. Orientation matrices are selected as the matrices that have enough group B and group C reflections to fulfill the completeness criteria.

To improve the approximation of the **U**-matrices, equation (3.3) is linearised around the found **U**-matrices and a least-squares fit is performed using the $\mathbf{G}_\omega$-vectors that belong to group B. With the new, refined **U**-matrices the $\mathbf{G}_{hkl}$-vectors in group C can be designated a $\mathbf{G}_\omega$-vector by choosing the option that deviates the least from the expected position according to equation (3.3).

3.5 Refinement and tomographic reconstruction

Once a first estimate of the **U**-matrices has been made, the orientations can be further refined and the grain shapes reconstructed. The necessary input data are the calculated **U**-matrices together with known parameters, such as the measurement geometry, the **B**-matrix, and the recorded diffraction peaks. Hence, the first refinement step is to define grain objects in Python from the **U**-matrices and **B**-matrix. The grain orientation is stored as the inverse of the matrix product between **U** and **B**, i.e $\mathbf{UB}^{-1}$. Peaks are assigned to the grain objects by using the $\mathbf{UB}^{-1}$-matrix to calculate the Miller indices that a given diffraction peak should have, and then comparing how close the result is to a vector of integers. If the norm of the difference between the exact result and the result rounded to the nearest integers is smaller than a user-specified tolerance, the peak is indexed to the grain. An extract from the flowchart found in appendix A which shows the parts of the s3dxrd package that are used for peak refinement can be seen in Figure 3.4.

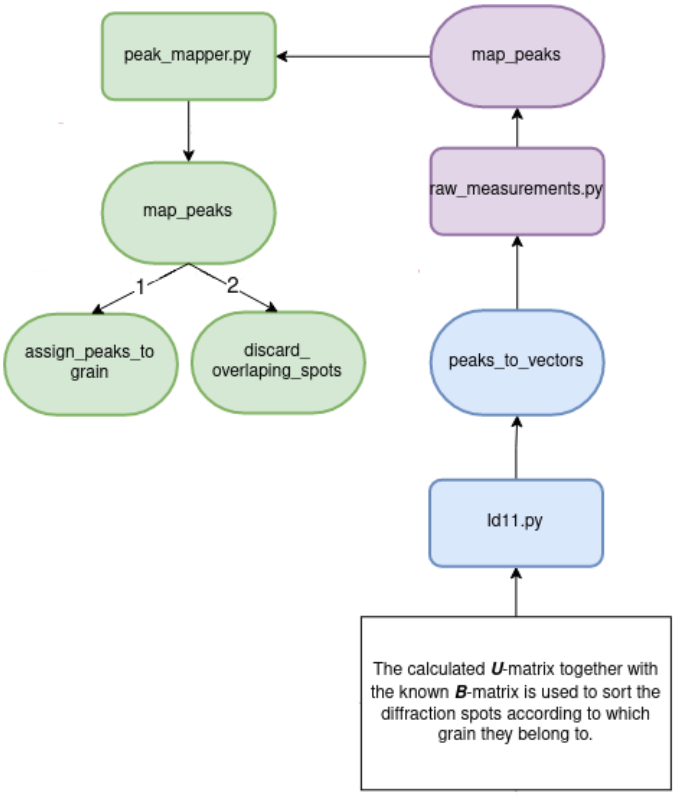


Figure 3.4: Extract from the flowchart in appendix A showing the relevant parts of the s3dxrd package used for peak refinement and assignment. White boxes shows a processing step, coloured boxes shows the connected files and functions in the s3dxrd package. Filenames are shown in rectangular coloured boxes, where a unique colour is designated to each file, and Python functions are shown in oval coloured boxes.

Using the diffraction peaks that have been assigned to a grain, a sinogram is created. Note that the only knowledge needed to create the sinogram is the recorded diffraction spots of the peaks assigned to the grain, along with the information on what coordinates the sample had when the diffraction spots were recorded. From the sinogram, filtered back-projection is performed to obtain the reconstructed grain shapes. The reconstructed grain shapes are thresholded since the intensity in the reconstructed grain map is of no relevance, and a binary map is sufficient for describing the grain shapes. Filtering is also performed to make sure that there are no missing pixels in an otherwise satisfactory grain reconstruction. Figure 3.5 shows the parts of the s3dxrd package that are used for grain shape reconstruction.

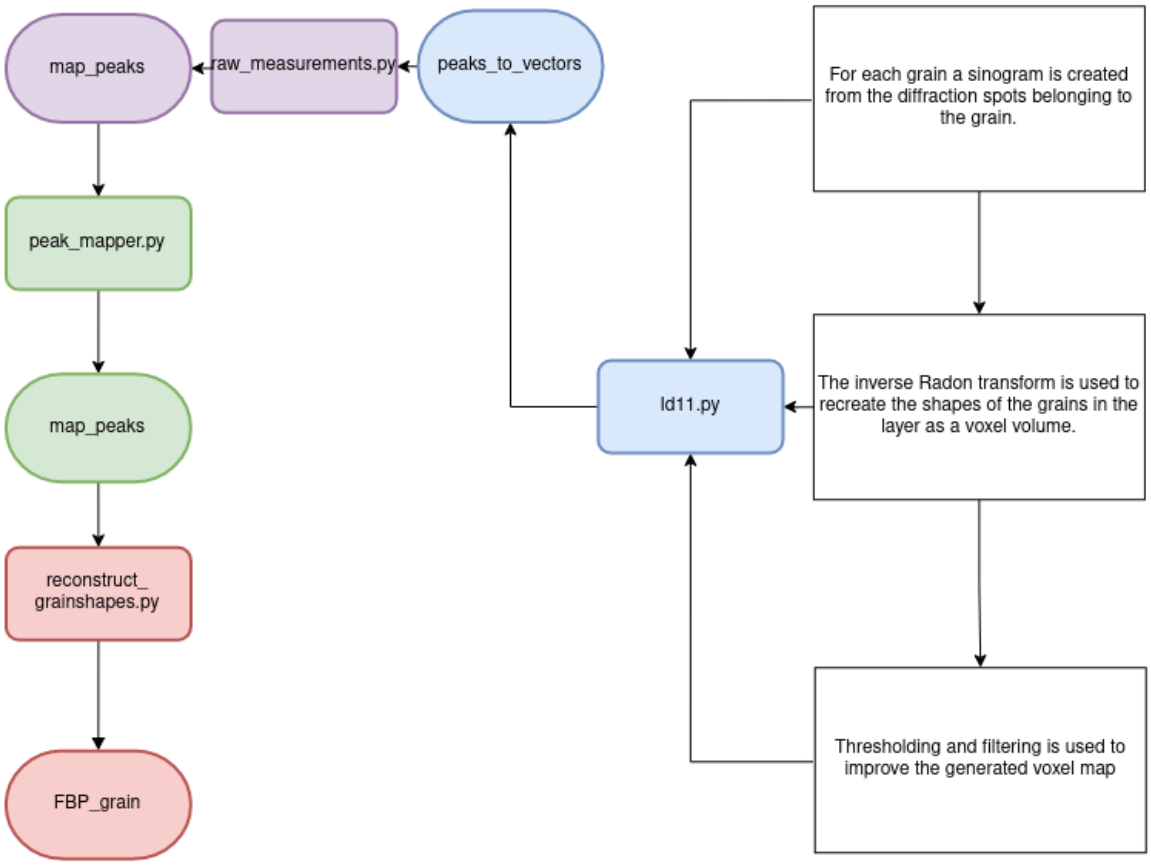


Figure 3.5: Extract from the flowchart in appendix A showing the relevant parts of the s3dxrd package used for grain shape reconstruction using filtered back-projection.

Once the grain shapes have been obtained, the center of mass of each grain can be calculated. Until this stage, it has been assumed that the scattered rays originated at the origin of the ω-coordinate system. When the grain shapes have been obtained, the scattered rays can be assumed to have originated at the grain centers of mass, and be directed to the centers of mass for the different diffraction spots. By recalculating the $\mathbf{G}_\omega$-vectors using this method, the diffraction peaks can be reassigned to the grains. Upon reassignment, the diffraction spots will be more accurately assigned to the grains.

Hence the $\mathbf{UB}^{-1}$-matrix can be refined as well, once it is known that not all diffraction spots are to be considered when trying to fit the grain orientation matrix. This procedure of calculating the centers of mass for the grains, updating the assigned peaks and updating the $\mathbf{UB}^{-1}$-matrices is divided into several parts performed by different scripts. See Figure 3.6 for details.

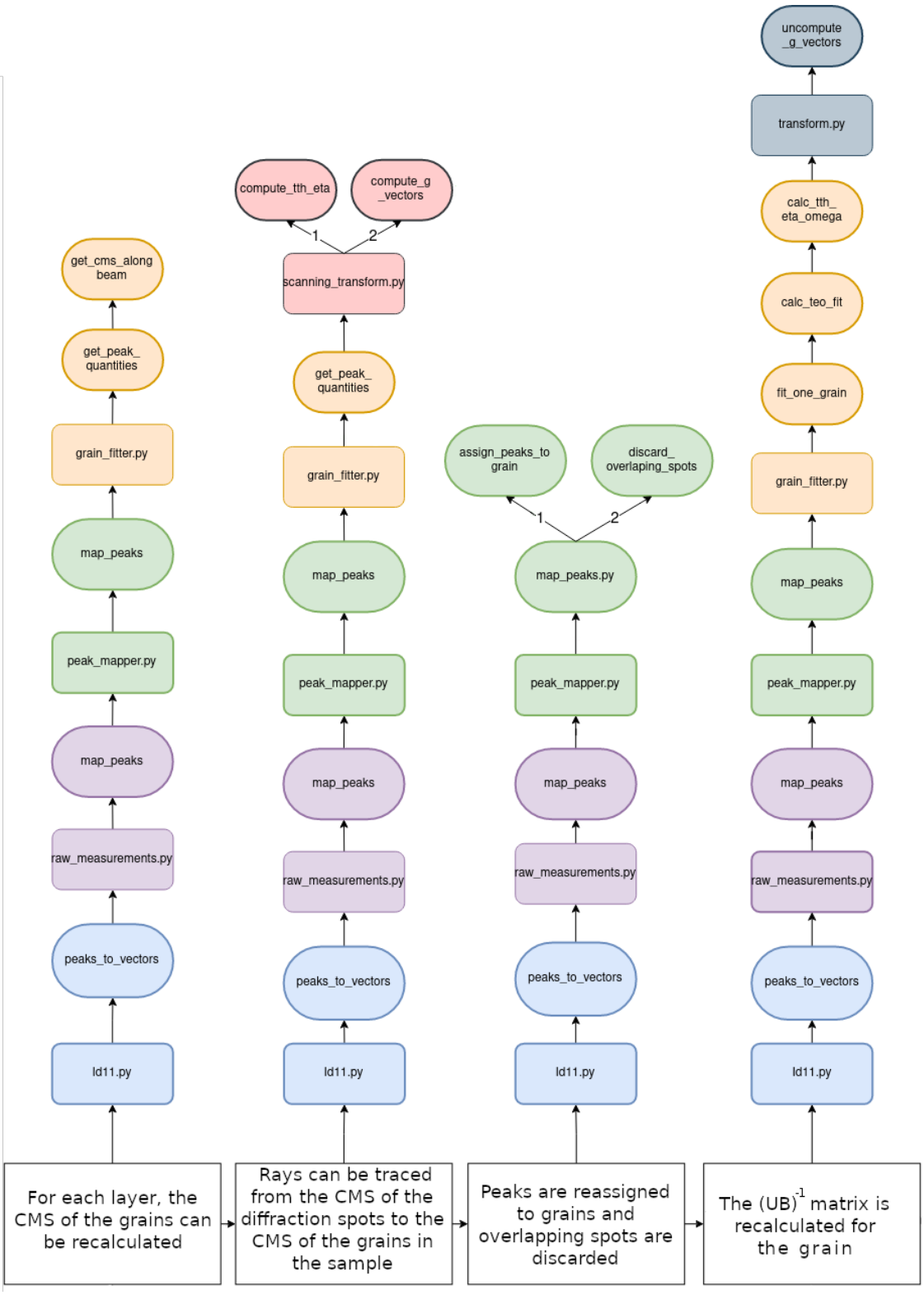


Figure 3.6: Extract from the flowchart in appendix A showing the relevant parts of the s3dxrd package for updating grain centers of mass, updating the calculated $\mathbf{G}_\omega$-vectors, reassigning peaks and updating the $\mathbf{UB}^{-1}$-matrices.

For the purpose of increasing grain reconstruction accuracy, the above described process is performed again, starting with the creation of the grain sinograms. One may repeat this process until the centers of mass of the reconstructed grains no longer change noticeably. In practice, this type of convergence is obtained after two iterations, hence the process is repeated twice, without checking any criterion of convergence. Once the diffraction spots have been assigned to grains and the grain shapes have been reconstructed to a satisfying degree of accuracy, quantities such as the Bragg angle, η-angle and $\mathbf{G}_l$-vectors can be updated for each diffraction spot belonging to a grain. Figure 3.7 shows how the parts of the s3dxrd package used for updating these quantities are connected.

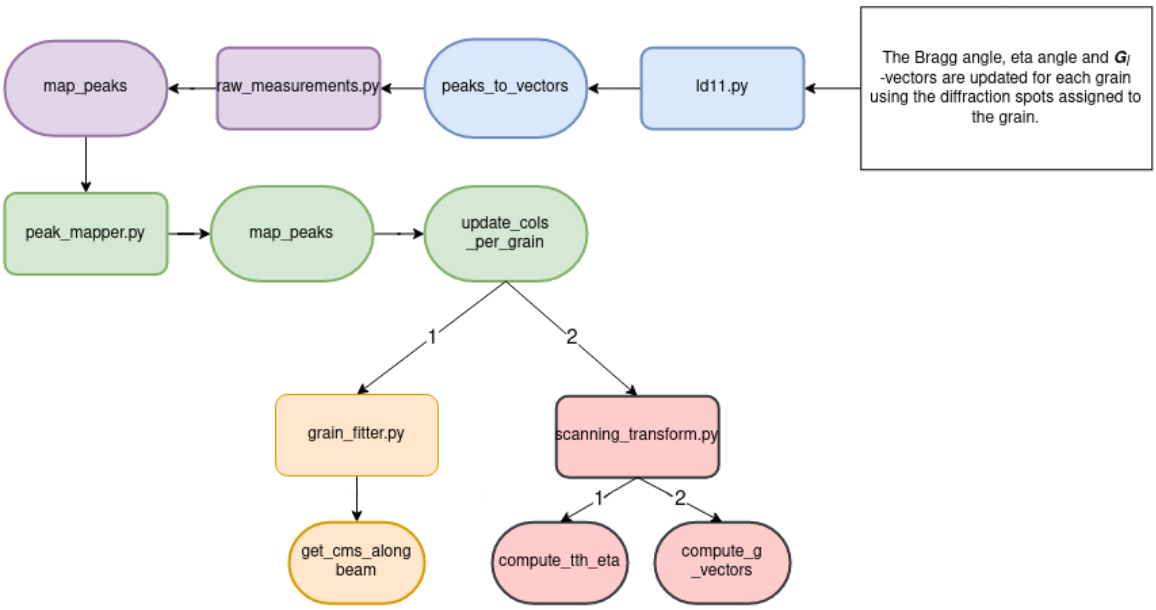


Figure 3.7: Extract from the flowchart in appendix A showing the relevant parts of the s3dxrd package for updating the Bragg angle, η-angle and $\mathbf{G}_l$-vectors.

Attention can now turn to assembling all of the grains in a layer to a slice covering a specific z-coordinate. For this, the grain reconstructions are calculated again, normalised and stored in a list. Since two grains may try to occupy the same area by partially intersecting each other, the binary map for the grain layer is created by assigning the pixels to the grain whose reconstruction has the highest intensity.

Once all grains have been assembled into a reconstruction of the entire layer, the reconstructed map is filtered to remove single pixels that are disconnected from any grains, and any missing pixels in a grain are filled in. All the reconstructed layers can now be assembled into a three-dimensional voxelated volume. Each voxel is given a number that signifies which grain the voxel belongs to. Where in the s3dxrd package the reconstruction of the different layers in the sample and conversion of the reconstructions to voxel maps is done can be seen in Figure 3.8.

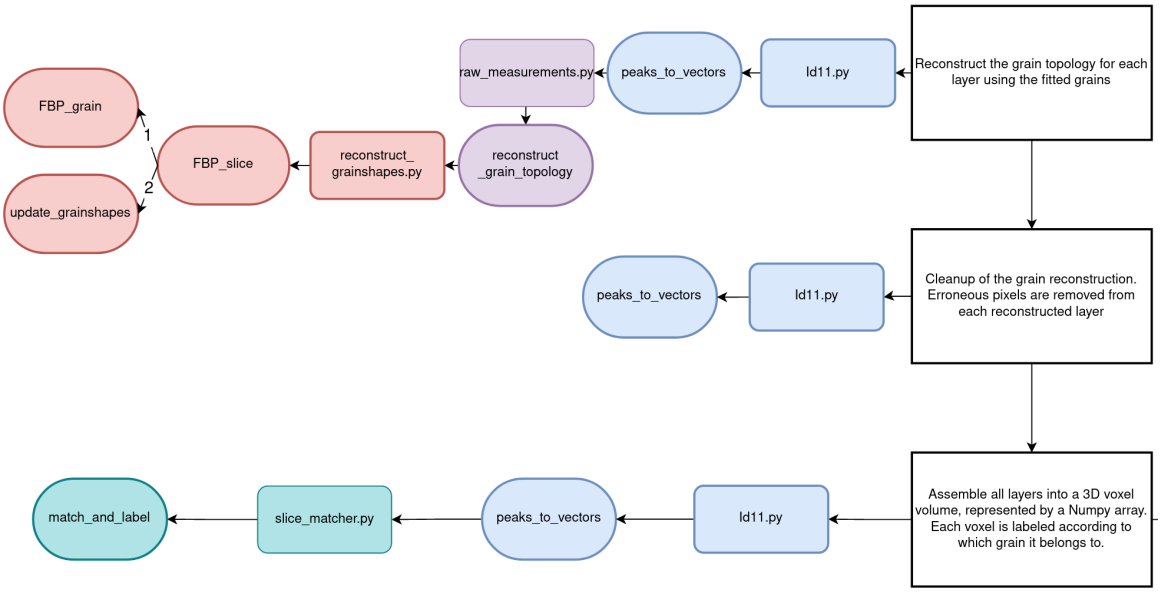


Figure 3.8: Extract from the flowchart in appendix A showing the relevant parts of the s3dxrd package for reconstruction of the layers in the sample and assembly into a voxel volume.

For every layer, the grains are converted into two-dimensional polygon objects. The average strains for each measured diffraction spot are calculated using equation (3.4) from the measured Bragg angles and a set of reference Bragg angles measured when the sample is without any applied load. A similar polygon representation is made of the x-ray beam itself. This allows for the intersections between the grain and the x-ray beam to be calculated to obtain the size of the area that was illuminated by the x-ray beam, the direction of the x-ray beam in the ω coordinate system and the positions where the beam entered or exited the grain. Any measurements that missed the sample are excluded from further processing. Where in the s3dxrd package this is done can be seen in Figure 3.9. The resulting quantities from the processing of the 3DXRD-data are saved to a file, for further processing using the ASR algorithm.

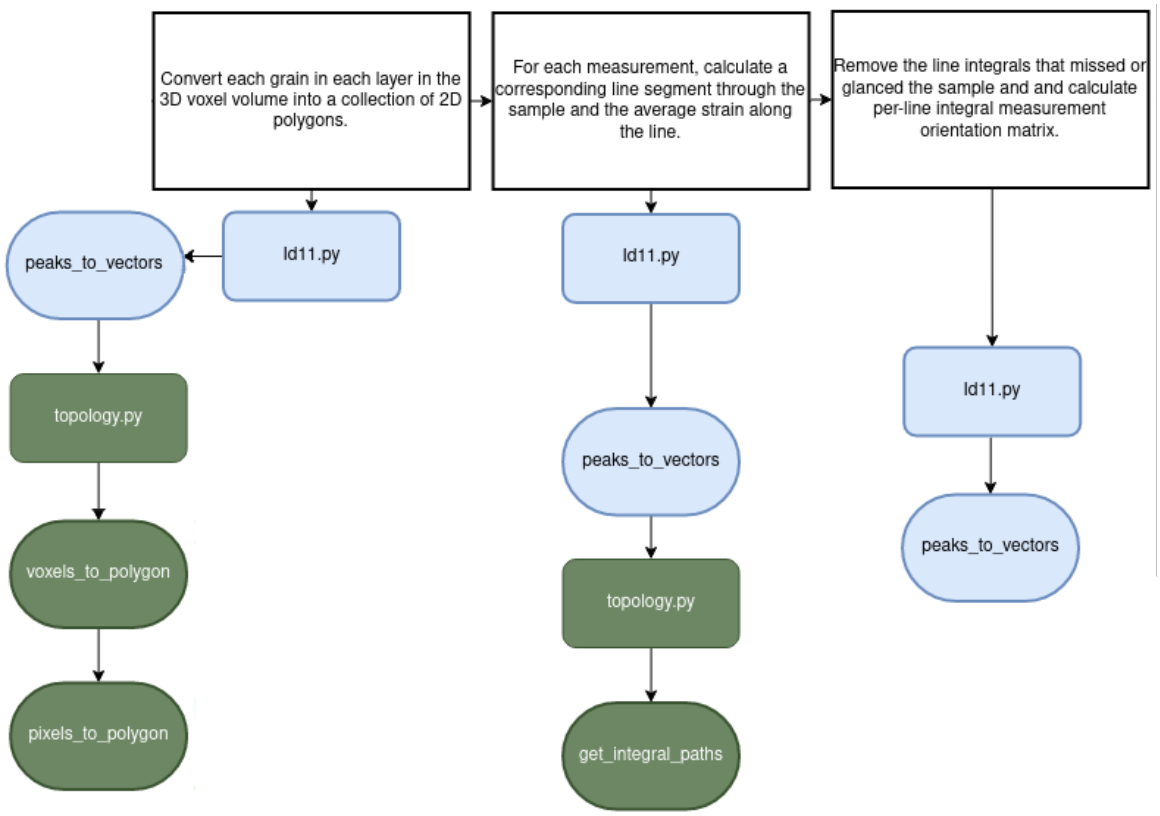


Figure 3.9: Extract from the flowchart in appendix A showing the parts of the s3dxrd package used for conversion of grain representations to polygon format, calculation of average strains and removal of erroneous line integrals.

3.6 Algebraic strain refinement

The processing of the data done so far only yields a set of average strains along the lines in the sample traced by the x-ray beam. If a three-dimensional map of the intra-granular strain variations is to be created, more data processing needs to be done. Algebraic strain refinement (ASR) is a method by Henningson *et al*[2] for converting the average strains and the voxelated grain reconstructions to a voxelated strain map showing the intra-granular strains as extrapolated from the measured average strains. The average measured strains are found from the difference in Bragg angle between the reflection from a crystal subjected to an applied load, versus the corresponding reflection from an unloaded reference state. Henningson *et al* [2] provides an expression for the measured average strain in a volume $\mathcal{R}$ illuminated by the x-ray beam based on the local strain tensor $\mathbf{E}_\omega$, the total illuminated volume V_{tot} and the strain direction $\bar{\mathbf{n}}_\omega$ as

$$\bar{\varepsilon}_m = \frac{1}{V_{tot}} \int_{\mathcal{R}} \bar{\mathbf{n}}_\omega^T \bar{\mathbf{E}}_\omega \bar{\mathbf{n}}_\omega dx_\omega dy_\omega dz_\omega. \tag{3.5}$$

Since the grains are represented as a voxelised structure, and a corresponding linear system of equations for describing the voxel strains is sought, Henningson *et al* derived a discrete form of equation (3.5) given as[2]

$$\bar{\varepsilon}_m \simeq \frac{1}{V_{tot}} \sum_i^N V_i \bar{\mathbf{n}}_\omega^T \bar{\mathbf{E}}_\omega \bar{\mathbf{n}}_\omega dx_\omega dy_\omega dz_\omega, \tag{3.6}$$

where V_i is the volume of voxel number i and N is the number of voxels illuminated by the beam for a given sample position. Equation number (3.6) applies to a single measured reflection, but can be extended to an entire line through the sample, containing, in total, a number of M different measurements.

Defining a matrix, $\mathbf{A}$, that is part of a system of equations,

$$\mathbf{As} = \mathbf{m}, \tag{3.7}$$

where each row in the system of equations corresponds to one measurement according to equation (3.6). The vector $\mathbf{m}$ contains the average strains for all measurements along a line in the grain and $\mathbf{s}$ contains the sought strains. Note that both $\mathbf{s}$ and $\mathbf{A}$ are matrices containing multiple other matrices, N matrices for $\mathbf{s}$ and $N \times M$ matrices for $\mathbf{A}$. The matrices $\mathbf{A}$, $\mathbf{s}$ and $\mathbf{m}$ are defined as follows[2].

$$\mathbf{A} = \begin{bmatrix} (V_1^1/V_{tot}^1)\mathbf{a}^1 & \dots & (V_N^1/V_{tot}^1)\mathbf{a}^1 \\ \vdots & \ddots & \vdots \\ (V_1^M/V_{tot}^M)\mathbf{a}^M & \dots & (V_N^M/V_{tot}^M)\mathbf{a}^M \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_N \end{bmatrix}, \quad \mathbf{m} = \begin{bmatrix} \bar{\varepsilon}_1 \\ \vdots \\ \bar{\varepsilon}_M \end{bmatrix}.$$

To describe the 3×3 strain components in the matrix $\mathbf{E}_\omega$ using the column matrices labeled ϵ_i Henningsson *et al* [2] applied Voigt notation. For the properties of Voigt notation and symmetries of the strain tensor, the reader is referred to textbooks in materials modeling and solid mechanics, for example Ristinmaa and Ottosen[12]. The vectors $\mathbf{a}$ and ϵ_i as given by Hennningson *et al*,[2] are

$$\mathbf{a} = \begin{bmatrix} n_1^2 & n_2^2 & n_3^2 & 2n_2n_3 & 2n_1n_3 & 2n_1n_2 \end{bmatrix},$$
$$\epsilon_i = \begin{bmatrix} E_{11} & E_{22} & E_{33} & E_{23} & E_{13} & E_{12} \end{bmatrix}^T.$$

Processing of the recovered data is performed on a per slice basis and starts with the conversion of the polygon grain representations in a slice to a two-dimensional mesh. When only a single grain is to be examined, the relevant data can be retrieved by using a mask, i.e an array of Boolean values where the value *true* at a given index means that the data for the corresponding index in the data set will be will be included in the strain calculations and the value *false* means that the data will be excluded. Since the data file used as input to the ASR algorithm contains the data ordered in dictionaries after both z-position and grain index, the relevant values for all slices, or just a single grain, can be retrieved.

The **A**-matrix is assembled row by row according to equation (3.6). The mesh is converted back into a polygon mesh, this time the polygon mesh is voxelated, rather than just containing the contours of the grain, allowing for the volume of intersection between the voxels of the grain and a polygonal representation of the x-ray beam to be calculated. In equation (3.6) this is represented by the volumes named V_i. The vectors **m** and **a** containing the average strains and their respective directions are given as input data to the ASR algorithm, although the strain directions are rewritten from three-component vector notation to Voigt notation. As can be seen in Figure 3.10 the processing of the 3DXRD data for intra-granular strain calculation is done in a different part of the s3dxrd package than the previous data processing.

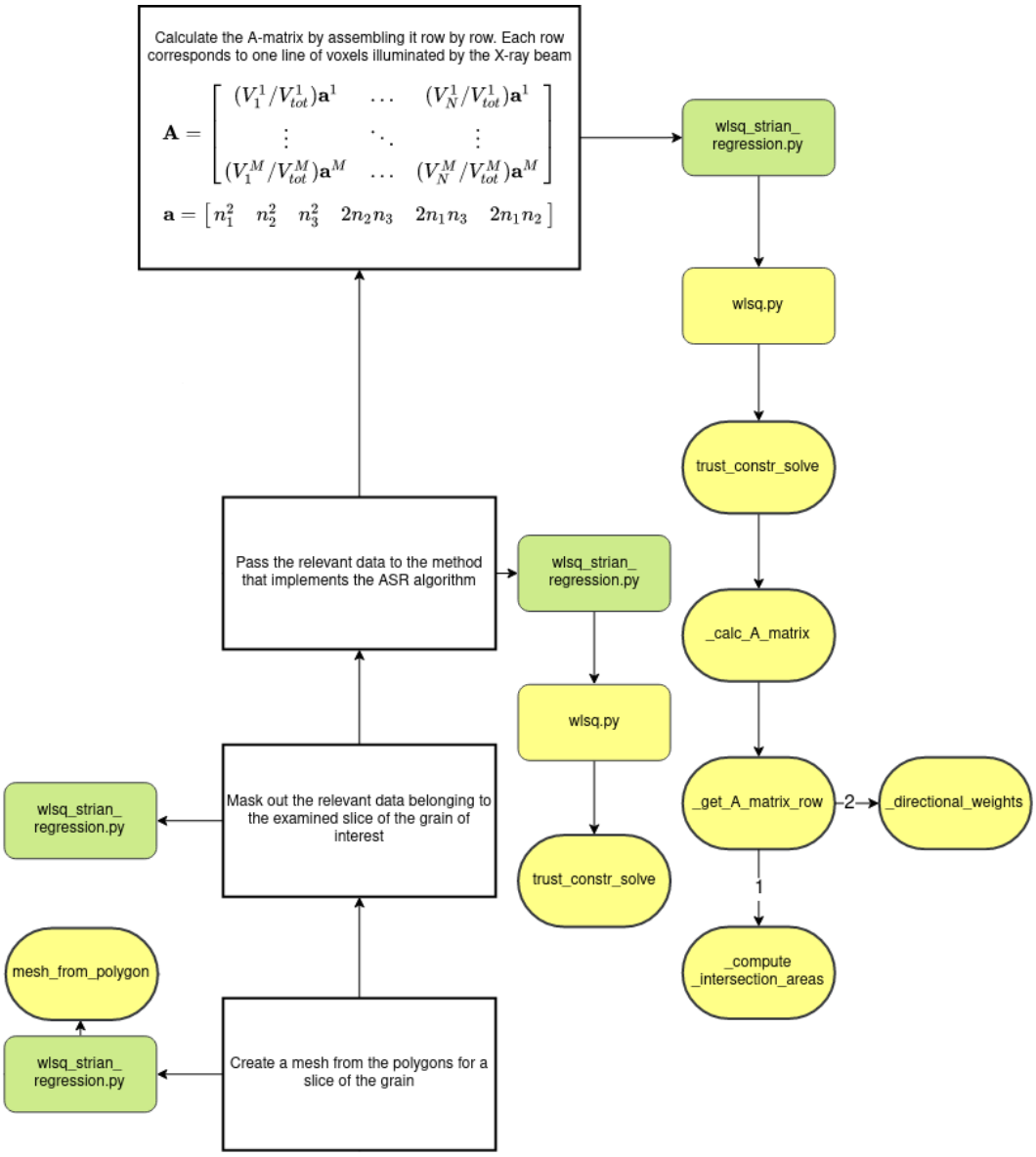


Figure 3.10: Extract from the flowchart found in appendix A showing which parts of the s3dxrd package that are used for conversion from mesh format to polygon format, where the ASR algorithm is located and where the **A** matrix is built.

Since the strain is presumed to vary smoothly from one voxel to the next, i.e no discontinuities, fractures or point loads are affecting the specimen, a constraint is imposed on the strain gradient. Henningson *et al* [2] formulates this constraint by bounding the maximum allowed difference between two voxels who share at least one corner. The specific size of the maximum allowed strain difference may be changed by the user and will be applied as the voxel strains are calculated. For the purpose of accounting for the varying measurement accuracy, Henningsson *et al*[2] introduced a set of weights to equation (3.7). These weights are chosen according to

$$w = \left(\Delta r \frac{\partial \bar{\varepsilon}_m}{\partial r} \right)^{-1}.$$

Henningson *et al*[2] refers to Borbely *et al*[13] and declares that that the value of Δr should be equal to 0.1 pixels due to low values of the measured diffraction angles. Figure 3.11 shows where in the s3dxrd package the constraints are implemented.

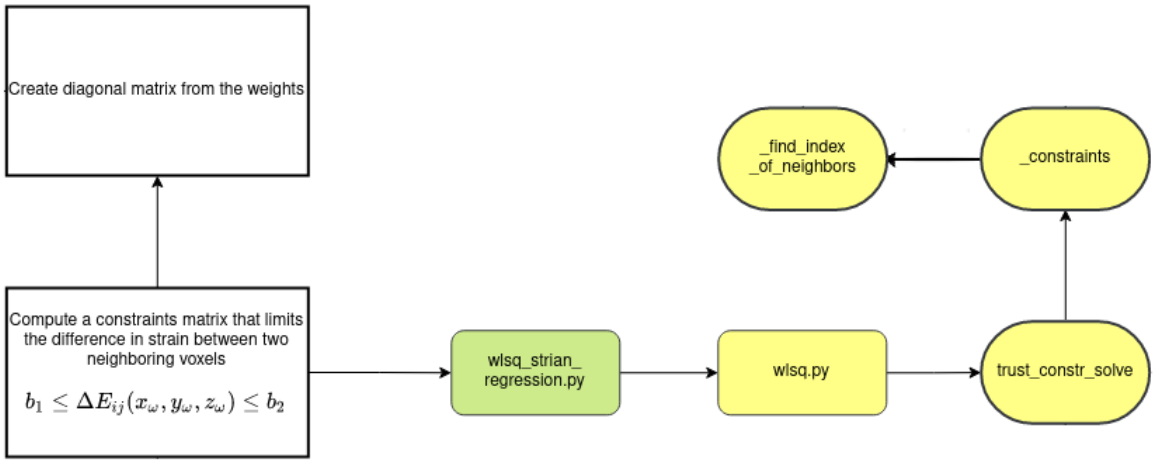


Figure 3.11: Extract from the flowchart found in appendix A showing which parts of the s3dxrd package that are used for calculation of the constraints matrix used for limiting the strain gradient between neighbouring voxels.

Defining a corresponding matrix of weights, $\mathbf{w}$ and multiplying both sides of equation (3.7) yields a weighted matrix equation.

$$\mathbf{wAs} = \mathbf{wm}. \tag{3.8}$$

Equation (3.8) can be solved in a least-squares sense by defining a cost function from equation (3.8) as[2] $\mathcal{A} = \frac{1}{2}||\mathbf{wAs} - \mathbf{wm}||_2^2$. From the weighted matrix equation, the Jacobian for the cost function can be derived, since it is only composed of the matrices $\mathbf{A}$, $\mathbf{w}$, $\mathbf{s}$ and $\mathbf{m}$. The fact that an explicit expression for the Jacobian can be derived means that a numerical approximation can be avoided. Figure 3.12 shows where in the s3dxrd package the cost function and corresponding Jacobian matrix are calculated. The resulting strain tensor field for the measurement is found as the $\mathbf{s}$-matrix that minimises the value of the cost function.

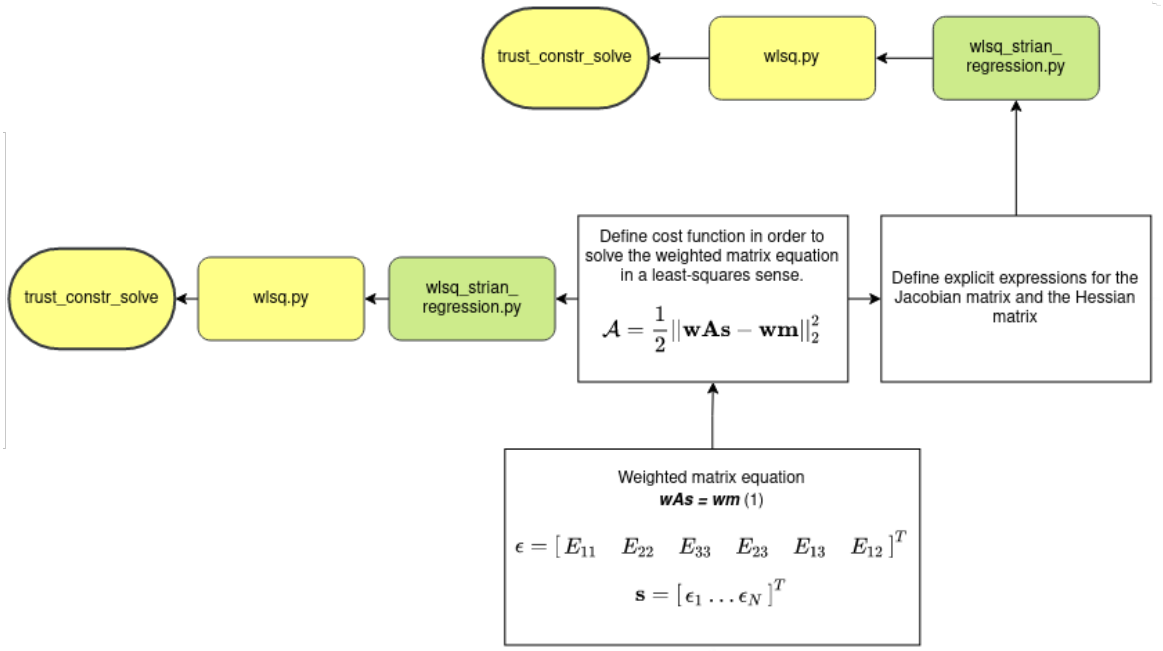


Figure 3.12: Extract from the flowchart found in appendix A showing which parts of the s3dxrd package that are used for defining the cost function from equation (3.8), and the parts that are used for calculating the Jacobian of the cost function.

Chapter 4

Additions and changes to the s3dxrd package

As part of the work done, a number of additions and changes of the s3dxrd package have been made. These changes are made in a separate branch of the s3dxrd package. At the time of writing it is not certain which of these contributions will be merged back into the s3dxrd package. Hence the following changes and additions should be considered as suggestions and are not decisive.

4.1 Changes to labelsparse.py

Two main changes have been made to labesparse.py. The script has been split up into different functions, including a main function, which contains the bulk of the code and previously was written just as a script. This allows for the code in labelsparse.py to be imported into another script without running all of the code in the file upon the import.

4.1.1 Added function: *sort_and_filter*

The function *sort_and_filter* has been added to the code. Since the number of recorded diffraction vectors is approximately 220000, each consisting of several diffraction spots, a considerable amount of memory is required to process all of them. Some of the recorded diffraction spots are also not mapped to any grain and need to be removed. Also, unsuitable η-values should be excluded from further processing. Hence there is a need for filtering of the input peaks. Using all peaks when performing the operations to recreate the strain tensor field takes several hours and, depending on the available computing power, it might not even be possible. Therefore it is desired to have the option to extract a subset of the data and only process that.

Of importance is also which peaks, and how many, are chosen for grain shape reconstruction and strain calculation, respectively. Recall that these two operations do not necessarily need to be performed on the same set of peaks, since all that is needed to create a tomographic reconstruction is the recorded diffraction spots, without any knowledge about which grains that the diffracted rays originate from. Note that in the algorithm, the diffraction peaks are in fact assigned to grains before the grain shapes are reconstructed, but to just get an image, this is strictly not necessary.

The strain reconstruction requires knowledge about how the peaks are distributed among the grains, but reconstructing the strains may be performed for a different set of peaks than the set used for reconstructing the grain shapes. This ability to use two sets of data is implemented in *sort_and_filter*, together with corresponding changes in other routines.

The functions needed for filtering the data set has been implemented previously. What was needed was a method for choosing a certain number of reflections per grain. The first stage in *sort_and_filter* is filtering of the η-values. This operation can be implemented in rather a concise way as seen in Figure 4.1.

```
# Filter high eta reflections
pks_recon.filter(np.abs(pks_recon.eta) < 175)
pks_recon.filter(np.abs(pks_recon.eta) > 5)
pks_strain.filter(np.abs(pks_strain.eta) < 175)
pks_strain.filter(np.abs(pks_strain.eta) > 5)
print("Peaks after removing unsuitable eta values: ", pks_strain.nrows)
```

Figure 4.1: Filtering of η-values can be done using previously implemented functions, as is done for the peak sets used for strain calculation and grain reconstruction.

For each grain object identified earlier, a mask is created that marks all peaks in the data set that belong to the grain in question. Identification of peaks is done using the same method of comparing how much the calculated Miller indices for a given diffraction peak deviate from being integer numbers when the orientation matrix of the examined grain is used. Iteration through all peak labels follows thereafter, where any peak that has been identified as belonging to the currently selected grain gets added to a dictionary as a value. The corresponding key will be the 2θ-value, i.e, two times the Bragg angle, recorded for the diffraction peak. Each grain will have its own dictionary of 2θ-values and the corresponding peak labels. The dictionary is then sorted using a built-in Python function, in such an order that the dictionary entries with the largest 2θ-values has the lowest indices in the dictionary. Figure 4.2 shows the assignment of peaks and the sorting of the dictionaries. Once the dictionary is sorted, the user may extract peaks based on the size of the 2θ-value in one of two possible ways.

```
    for i, gr in enumerate(gl):
        # Append a new dictionary to the grain list. Keys are the tth-values and the values are the peak labels.
        grainlist.append({})
        # Calculate the possible Miller indices for the grain.
        hkl = np.dot(gr.ubi, gve.T)
        # Round the Miller indices to integers.
        hkli = np.round(hkl)
        # Calculate the difference between the original hkls and the rounded hkls.
        drlv = hkli - hkl
        # Square the difference for each hkl set.
        drlv2 = (drlv * drlv).sum(axis=0)
        # If the difference between the original hkls and the rounded hkls is smaller than a tolerance, the hkls are
        # assigned to the grain. Note that all hkls are checked simultaneously, and hence belongs_to_grain is a list.
        belongs_to_grain = drlv2 < tolerance * tolerance

        for j in range(size):
            # If a hkl belongs to a grain...
            if belongs_to_grain[j]:
                # Set the corresponding tth and peak label as a key-value pair in the dict beloning to the grain.
                grainlist[i][pks_recon.tth[j]] = j

        # Sort the dict so that the largest tth-values come first.
        sort_largest_first = sorted(grainlist[i].keys(), reverse=True)
```

Figure 4.2: Assignment of peaks to different grains is done by comparing Miller indices. Peaks that are assigned to a given grain are stored in a dictionary together with their 2θ-value.

The first method for selection of the diffraction peaks is to choose the N_l largest 2θ-values and the N_s smallest 2θ-values. The other option is to find the mean 2θ-value and to choose N_l values larger than the mean value and N_s values smaller than the mean value. If the option to use the mean value is chosen, the first and the last value in the dictionary is used to calculate the mean value. Then the dictionary is traversed in the direction of decreasing 2θ-values until a value smaller than the mean value is found.

The found value will be slightly smaller than the mean value, but the value that is closest to the mean value might be the value that comes just before the found value in the dictionary, i.e a value slightly larger than the mean value. A comparison is made to find out which of the above situations that apply, and the value closest to the calculated mean value is used as starting point when choosing the values that will be used for grain shape reconstruction. The diffraction peaks used for strain reconstruction will always be chosen as the peaks with the N_s-largest 2θ-values. Do note that N_l, N_s and N_m are chosen by the user. The process for finding the mean value in the list of sorted dictionaries and appending the selected number of peaks, starting from the peak which has the 2θ-value closest to the mean value can be seen in Figure 4.3.

```
            for j, elem in enumerate(sort_largest_first):
                # Since the list is sorted, once an element is found that is equal to or smaller than the mean value,
                # the correct index for the value in the list that is closest to the mean value will be either the
                # current index, or the previous index.
                if elem <= mid:
                    # Calculate the difference between the mean value and the values found at the current place and
                    # previous place in the list.
                    diff1 = abs(sort_largest_first[j] - mid)
                    diff2 = abs(sort_largest_first[j - 1] - mid)
                    # The size of the error determines which index that is chosen.
                    if diff2 < diff1:
                        midind = j - 1
                    else:
                        midind = j
                    break
            # Append values starting from the index of the mean value and moving towards smaller values.
            for key in sort_largest_first[midind:midind + smallest]:
                filtered_values_recon.append(grainlist[i][key])
            # Append values starting from the index of the mean value and moving towards larger values.
            for key in sort_largest_first[midind - largest:midind]:
                filtered_values_recon.append(grainlist[i][key])
```

Figure 4.3: The 2θ-value closest to the mean value of all recorded 2θ-values is found by iterating through all dictionaries. Starting from the value found as a result of the iteration, peaks are appended to a list for grain shape reconstruction.

4.1.2 Removal of unindexed peaks

One of the operations that are performed within the function named *main* of labelsparse.py is indexing of the diffraction peaks. This means that all the diffraction peaks are given a label, corresponding to the grain that the diffraction peak belongs to. However, depending on which tolerance is chosen for indexing the diffraction peaks, not every peak may be assigned to a grain. These peaks need to be removed from further processing since they cannot be used for strain reconstruction and they will not be used to reconstruct the grains either. All peaks start with a label of -1, and upon assignment to a grain, these labels will be changed to a non-negative integer corresponding to the label of the grain. Any peak that has retained the label -1 after attempting to assign the peak to a grain could not be fit within the given tolerance, and is removed. The code for removing non-indexed diffraction peaks can be seen in Figure 4.4.

```
    # Remove spots that could not be indexed
    to_keep = [True] * np.size(idx, axis=1)
    nbr_removed = 0

    for i in range(np.size(order)):
        if idx[5, order[i]] == -1:
            to_keep[order[i]] = False
            nbr_removed += 1
        else:
            break

    print("To be removed due to not being associated with a grain: ", nbr_removed, " elements")
```

Figure 4.4: A short section of code is dedicated to removing peaks that could not be indexed to any grain.

4.2 Changes to Id11.py

The file Id.py contains the base of the routines necessary to convert the diffraction peaks to the vector format used for strain reconstruction. A significant amount of the data processing begins here and is then delegated to other functions and scripts. The changes here mostly consist of splitting what used to be one single function into multiple functions for ease of comprehension and increased modularity for other users. The functions that are used to organize the code, which previously were located entirely in *peaks_to_vectors*, are *map_and_recon*, *cross_slice_map*, *polygon_representation* and *parametric_integrals*.

4.2.1 Selection of grain orientation matrices from initial estimations

The semi-raw data obtained from the 3DXRD-measurements contains a file with initial guesses for one $\mathbf{UB}^{-1}$-matrix per grain in the sample. Since the grain objects used for data organization are created based on the supplied $\mathbf{UB}^{-1}$-matrices, and one grain object is created per $\mathbf{UB}^{-1}$-matrix, the number of $\mathbf{UB}^{-1}$-matrices for a given slice has to agree with the number of grains that actually exist in the slice. If too many $\mathbf{UB}^{-1}$-matrices are supplied, the algorithm would attempt to recreate the grain shapes of grains that do not exist in the slice and attempt to assign peaks to these grains.

It is emphasized that all the supplied $\mathbf{UB}^{-1}$-matrices do belong to some grain in the collection, but not all grains are present in every slice. Fortunately one may study how many peaks get assigned to the different grains in a slice. Grains with $\mathbf{UB}^{-1}$-matrices that do not correspond to a grain from the slice should get few peaks assigned to them, and these grain objects can be removed. Due to the fact that the indexing of diffraction peaks to grains is based on the number of available $\mathbf{UB}^{-1}$-matrices, all the peaks have to be reassigned to the remaining grains to avoid discontinuities in the enumeration of the grains, e.g., the grains being labeled 0, 3, 4 and 5 due to the removal of grains 1 and 2. This would make iterating through the grain indices more difficult.

The criterion for a grain to be considered as being present in a slice is based on the largest number of peaks that gets assigned to any of the grains present before any grains are removed. If any grain gets less peaks assigned to it than 10 % of the number of peaks assigned to the grain with the most peaks, it is considered to be non-present in the slice and is removed from the list of grain objects. The cutoff at 10 % is rather arbitrarily chosen, but has been seen to remove the unwanted grain objects and only leave behind grain objects that correspond to grains actually present in the slice. The implementation of the algorithm that removes excess grains can be seen in Figure 4.5.

```
    for i in range(2):
        labels = np.zeros(colf.nrows,
                          'i') - 1  # Initiate a list of reflection pixel labels. The list contains one element for each
        # for eagh pixel that recorded a reflection. The value of the element ranges from -1 to one less than the
        # number of grains, i.e the first grain will be labeled 0.
        drlv2 = np.ones(colf.nrows,
                        'd')  # Vector for storing the difference between the measured hkl and the hkl rounded
        # to the nearest integers.
        tol = 0.075  # Tolerance for assigning reflections to as grain.

        assigned_to_grain = [0] * len(gl)

        for ii, ubi in enumerate([g.ubi for g in gl]):
            nbr_assigned = cImageD11.score_and_assign(ubi, gve, tol, drlv2, labels, ii)  # Grain labels are updated here
            assigned_to_grain[ii] = nbr_assigned
            print("Peaks assigned to grain nbr ", ii, " :", nbr_assigned)

        if i == 0:  # This is only to be done once.
            grains_to_remove = []
            grains_to_keep = []
            for jj, peaks in enumerate(assigned_to_grain):
                if peaks > 0.1 * np.max(assigned_to_grain):
                    grains_to_keep.append(gl[jj])
                else:
                    grains_to_remove.append(jj)

            gl = grains_to_keep
            print("Grains nbr ", grains_to_remove, " recieved less than 10% of the maximum number of "
                                                   "assigned peaks to a grain and has been deleted")
```

Figure 4.5: The initial list of grains in a slice contains all grains in the entire sample. The code seen here counts how many peaks that have been assigned to the grains. Any grains with significantly less peaks relative to the grain with the most peaks are removed.

4.3 New script: assemble.py

The entire procedure of labeling the diffraction spots, identifying which pixels belong to the same diffraction vector, processing the diffraction vectors and converting them to a vector file format and calculating the intra-granular strains using the ASR method is all controlled from the file named assemble.py. Here, the user may choose to run labelsparse.py, to_vectors.py and reconstruct_all.py or a combination of these. Dividing the computations into different functions that may or may not be executed has the advantage that any changes to the code can be tested without performing the entire process from diffraction peak indexing to strain reconstruction.

4.3.1 Added function: *label*

The function named *label* creates a list of all the files containing the diffraction data by appending the file paths to a list. Two different loops are used for this, one treating the files numbered zero to nine, another loop appends the filenames of file number 10 to file number 45. For each file in the list, the code in the *main*-function of labelsparse.py is performed and the corresponding peaks used for reconstructing the grain shapes, the peaks that will be used for strain reconstruction and the refined $\mathbf{UB}^{-1}$-matrices are stored in files together with the original data for the slice at a location specified in labelsparse.py. The files containing measurement parameters and the initial guesses for the $\mathbf{UB}^{-1}$-matrices are never altered, hence their location is only specified once in the code.

4.3.2 Added function: *vectorize*

In the function named *vectorize* the files containing the peaks that will be used for reconstructing the grain shapes, the peaks that will be used to calculate the intra-granular strains and the $\mathbf{UB}^{-1}$-matrices that correspond to a given slice are collected in lists. Much in the same way as for *label*, the file paths are stored in the lists and not the data. The data is then processed using the algorithms in to_vectors.py to obtain the grain shape reconstructions, average strains, etc. The data are split into two different files from a complete data set which contains in total 46 different slices. Therefore the *main* function of to_vectors.py is called twice, once for all data in slices number 0 to 22, and again for all data in slices 23 to 45.

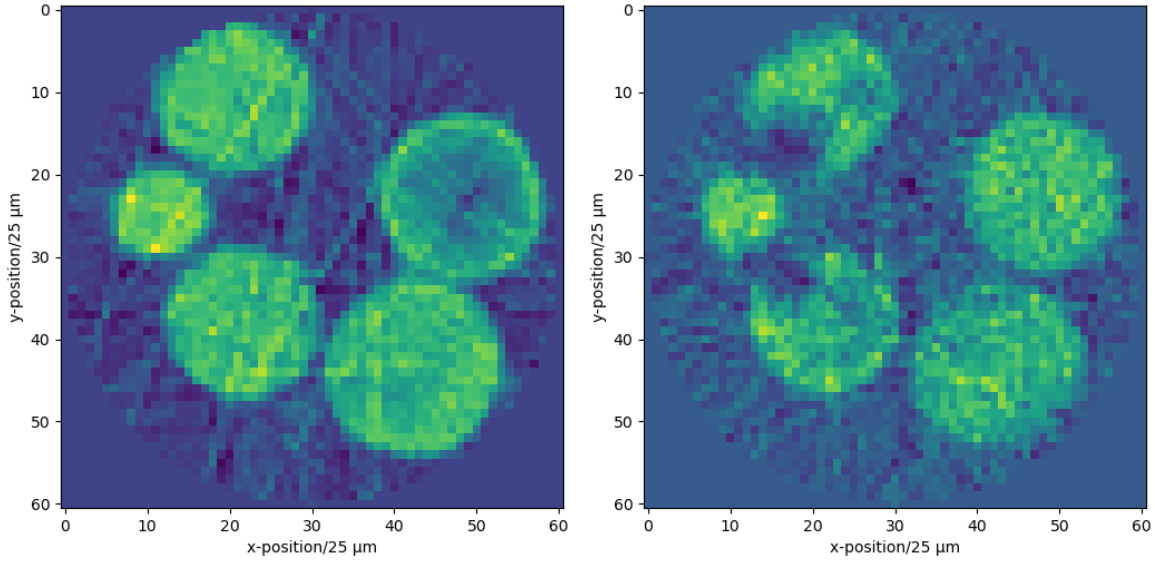
4.3.3 Added functions: *reconstruct* and *main*

The function reconstruct simply exists for the purpose of being able to control whether or not a reconstruction of the intra-granular strains is performed. Hence it is rather short and only consists of reading the previously created vector files and passing the data to the function named *main* in reconstruct_all.py. The function named *main* in assemble.py is used to specify what functions in assemble.py that should be used, and therefore only consists of calls to the other functions in the script.

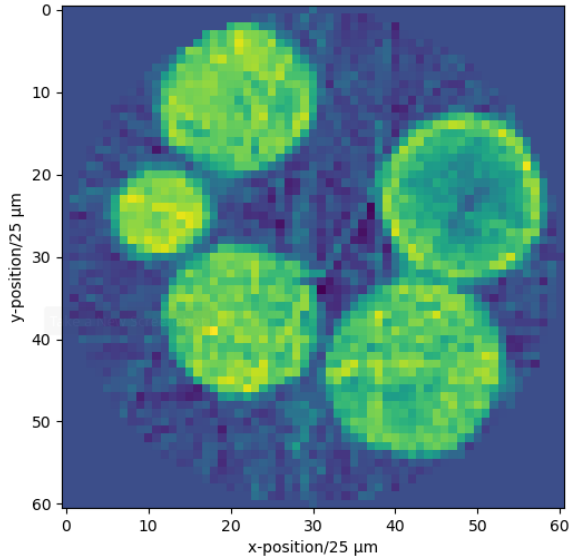
Chapter 5

Results and findings

Results from the measurements of the grain assembly when under load and other results from the examination of the data are presented here. Since the data available covers all strain components in the entire grain assembly, no single image can convey all information completely. If the reader is interested in obtaining the simulation results in their entirety, it is advised to contact the author. No separate chapter will be dedicated to discussing the results obtained. Instead, findings will be presented together with the relevant figures. Figures 5.2 to 5.5 have been generated using ParaView[14].



(a) (b)



(c)

Figure 5.1: The result of reconstructing the grain shapes using the method of filtered back-projection for different numbers of recorded diffraction peaks. Figure 5.1a shows the result when the peaks with the 5000 smallest 2θ values are used, Figure 5.1b shows the result for the 5000 peaks with the largest 2θ values. Figure 5.1c shows the result of the use of the peaks with the 5000 smallest 2θ values and the 5000 largest 2θ values.

Figure 5.1 reveals the importance of having a sufficiently large number of peaks for reconstructing the grain shapes. The effect is especially obvious in Figure 5.1b where holes appear in the grains due to an insufficient amount of reflections originating from these regions.

Compare Figure 5.1b to Figure 5.1a, where the grains which are incompletely reconstructed in Figure 5.1b are reconstructed to a much better quality. This implies that the angular distribution of the diffraction vectors is not uniform across all grain surfaces. Instead diffraction vectors with large 2θ-values seem to be located at certain areas in some grains, other grains do indeed seem to have a uniform distribution of diffraction vectors with large 2θ-values. Why the diffraction vectors are unevenly distributed across the grains when sorted after 2θ-angle remains unknown. The local deformations of the grains do change the 2θ angle when compared to a non-deformed reference state as seen in equation (3.4), but these changes are presumed to be too small to cause the uneven reconstructions seen in Figure 5.1b. In practice, the challenge of obtaining a reconstruction of good quality is solved by simply increasing the number of peaks used. The data contains approximately 45000 peaks per grain and slice, which is sufficient for obtaining accurate reconstructions.

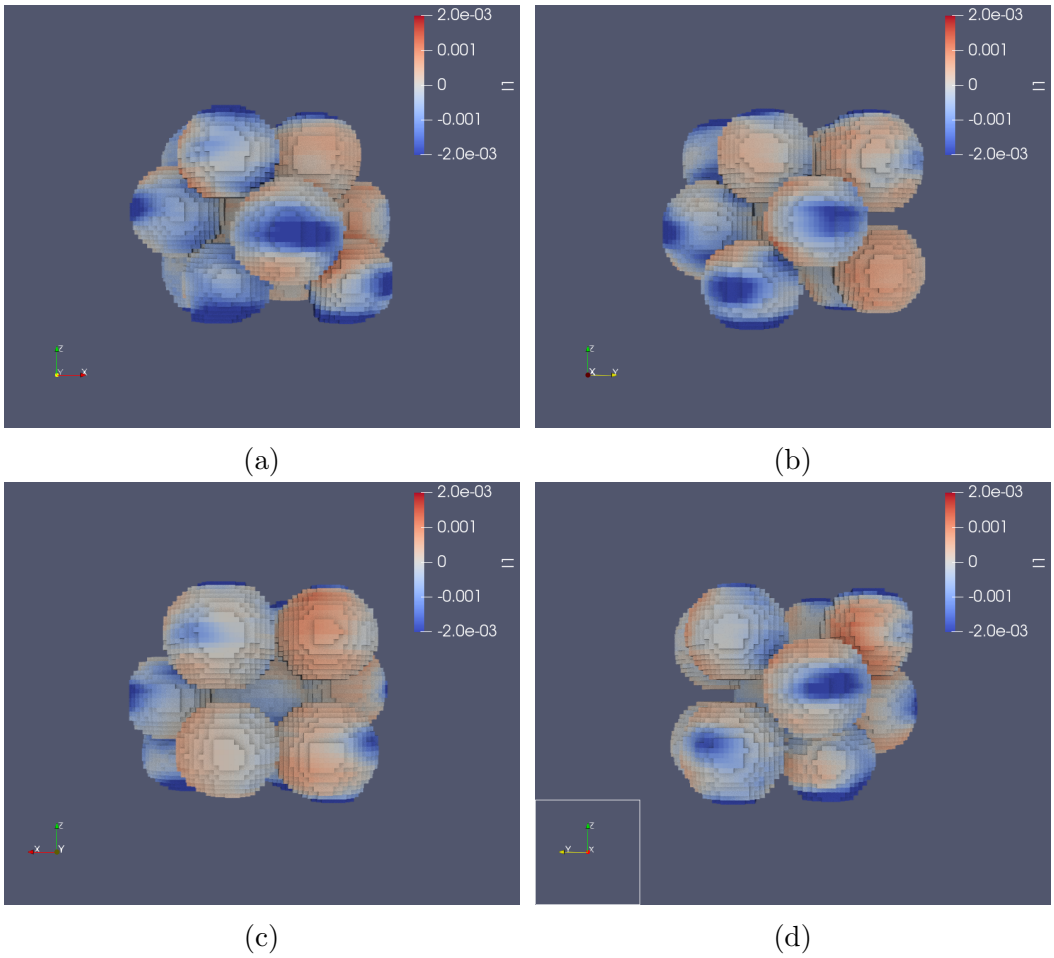
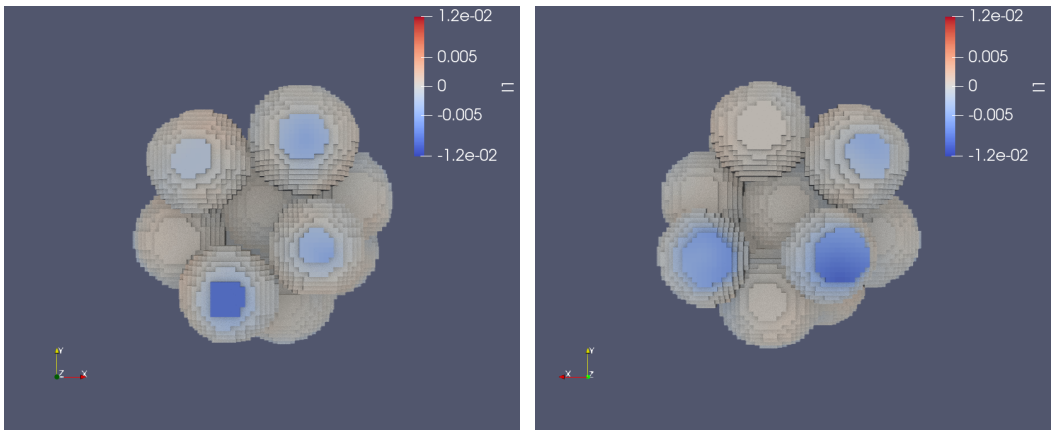


Figure 5.2



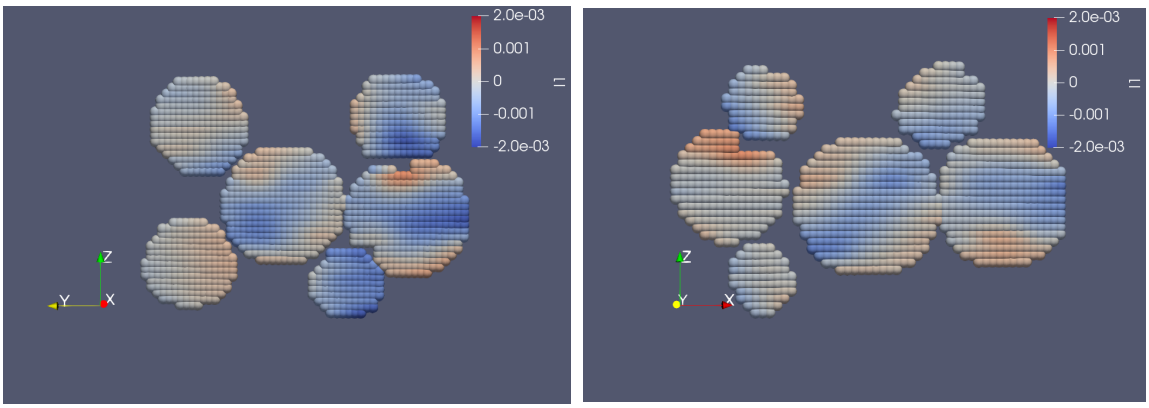
(e) (f)

Figure 5.2: Reconstruction of diffraction data obtained from 3DXRD-measurements and analysed using the ASR algorithm. The grain assembly is coloured based on the reconstructed volumetric strain and is oriented according to the coordinate system shown in the subfigures. Note that the scale of the colorbar may differ between subfigures.

Figure 5.2 shows the volumetric strain at the surface of the entire grain assembly. Note the areas in compression on the sides of the grains of Figure 5.2b. These areas resemble features seen when modeling similar systems of grains using the finite element method. As explained by Vestin [15] one possible reason for these types of compressive strains to occur is contact between the grain and the sides of the structure which contains the grains.

It is worth noting that one may also spot the areas on the grains that are in contact with the top and bottom of the containing structure, as measured along the z-axis. This can be noted in Figure 5.2b where some of the largest compressive strains can be found on the top and bottom of the grains, indicating that the surfaces are significantly deformed in comparison to other areas of the grains.

Figures 5.2e and 5.2f give slightly more insight into what is happening to the grains. Figure 5.2e suggests that one of the grains protrudes slightly more towards the positive z-direction than the other grains, and hence it is subject to a larger force from the piston pushing down on the grain assembly than the adjacent grains. This is indicated by the fact that the top layer of the bottom left grain experiences a noticeably larger strain compared to the top surfaces of the surrounding grains, even though they have contact areas of similar sizes. Figure 5.2f shows a similar situation, but this time it is the bottom right grain in the figure that appears to contact the bottom surface of the container the most.



(a) (b)

Figure 5.3: Two slices of the grain assembly from Figure 5.2 taken across the different z-layers, but oriented orthogonally relative each other. The orientation of the slices is given by the coordinate systems in the subfigures. The color of the slice represents the reconstruction of the volumetric strain, obtained by analysing data from 3DXRD-measurements with the ASR algorithm.

By studying the slices seen in Figure 5.3 conclusions can be made about the plausibility of the constraints used when calculating the strains. It is worth reminding of the fact that the strains are calculated independently for each layer in the z-direction and that the constraint which limits the strain gradient does not apply across different layers. Nevertheless it is noted in Figure 5.3 that the volumetric strain varies smoothly across the slices despite the fact that no constraint applies in the z-direction. That this is the case supports the plausibility of the results, since the strain is expected to vary continuously in all directions. Another feature, seen in Figure 5.3, aŕe the intra-granular contact surfaces. Some grains are in a state of compression across grain contact surfaces, indicated by blue areas on both sides of the contact region. This would correspond to the grains pushing against each other. However, some contact regions are characterised by one grain being in a state of compression, and the other grain being in a state of expansion. This would indicate some sort of shearing motion rather than the grains just being pushed towards each other.

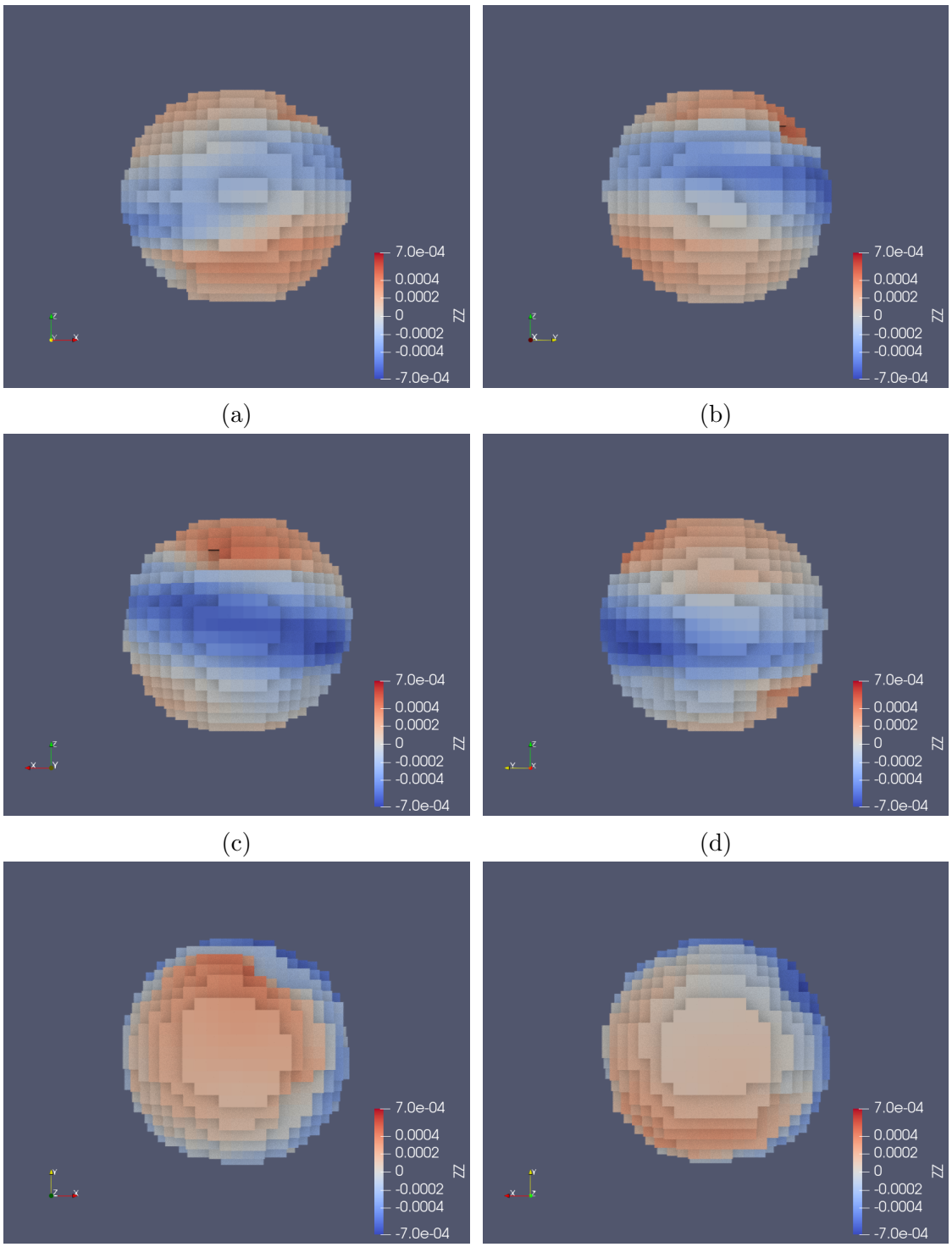
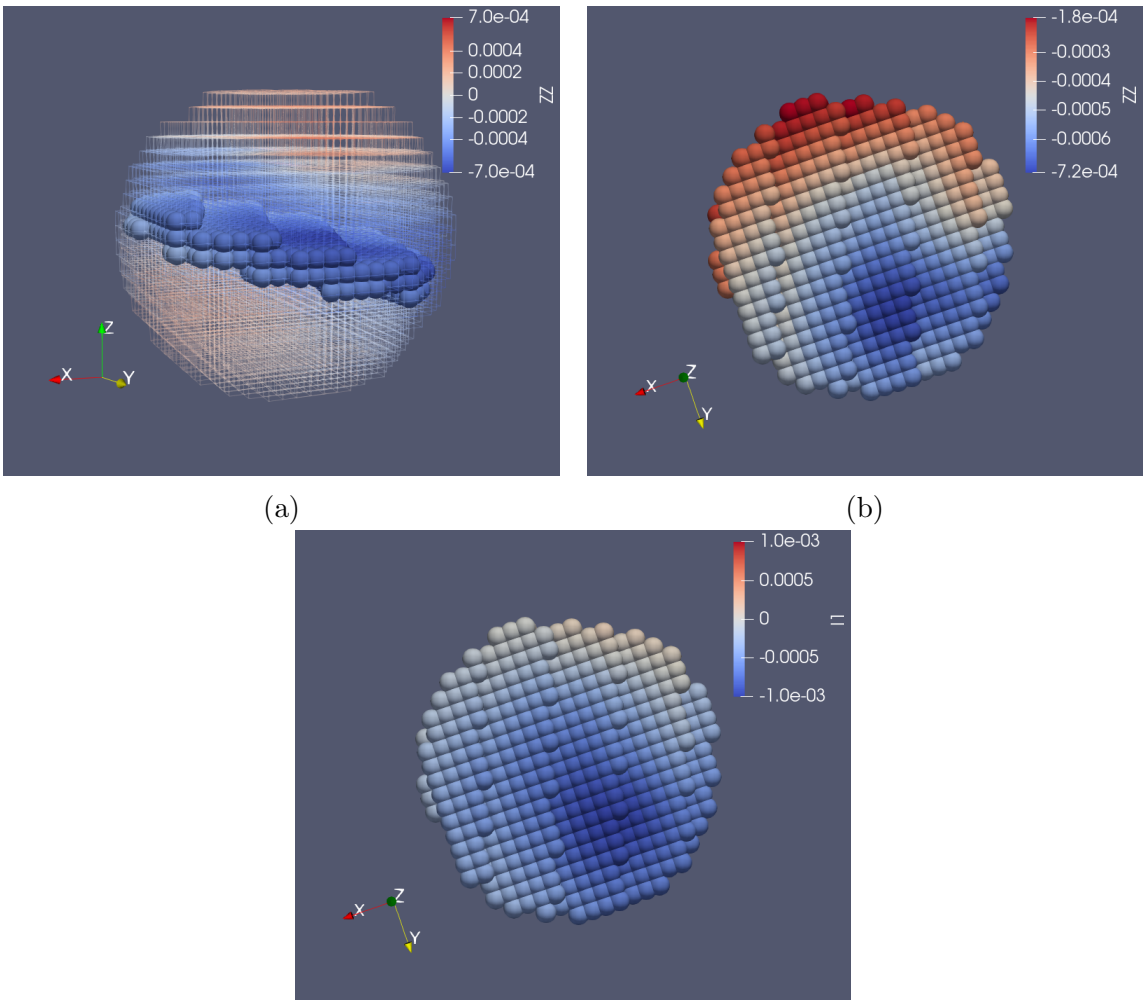


Figure 5.4: The center grain of the grain assembly from Figure 5.2 shown at directions given by the coordinate system in the subfigures. The grain color represents the magnitude of the zz-strain component, obtained from reconstruction of 3DXRD data using the ASR algorithm.



(a) (b)

(c)

Figure 5.5: The center grain with a slice of the grain highlighted. Figure 5.5a shows the grain with reduced opacity for the purpose of being able to locate where in the grain the slices in figures 5.5b and 5.5c are placed. Figure 5.5b shows the zz-strain component in the slice from Figure 5.5a and Figure 5.5b shows the volumetric strain in the slice. Note that although the color scheme is the same for all subfigures, the scale of the colorbar changes.

The middle grain seen from different angles in Figure 5.4 exhibits a rather interesting strain distribution when the zz-component of the strain is studied. As can be seen in Figure 5.4, the strain is negative in a ribbon all around the grain and positive elsewhere. Further investigation by extracting a slice of the relevant area, as in Figure 5.5, reveals that the zz-component of the strain is negative throughout the entire slice, as seen in Figure 5.5b. The fact that the behaviour of the strain is not just located to the surface of the grain suggests that the entire grain will deform along this plane. One option is that the grain is fracturing at this location. This is supported by the fact that reports by S.Hall suggests that this grain did indeed fracture during the experiment.

Chapter 6

Conclusions and outlook

6.1 Conclusions

In conclusion, this Bachelor's project has largely been about understanding three-dimensional x-ray diffraction, how the data are gathered and what information that can be extracted from the data. Novel aspects of the work include the implementation of algorithms to assemble the data from an entire scanned volume and the study of the obtained results, giving an insight into how a system of grains behaves when subject to an external load. It was shown that imposing a condition on the strain gradient in one plane yields a smoothly varying strain in a direction perpendicular to the plane subject to the constraint and the observed strains gave an insight into the behaviour of the grains when under load. The importance of having sufficient data when reconstructing the grains was also shown.

6.2 Outlook

The results obtained spark a number of new questions that are outside the scope of this project. Regarding the center grain shown in Figure 5.4, it would be of interest to see how the grain responds to an increased load. In fact, it would be useful to perform the analysis done here for different load levels and thereby obtain an understanding for how the system responds as the load varies. Since data collection was done by increasing the external load and allowing the system to reach a state of equilibrium before the scanning was performed the necessary data has been gathered, but is yet to be processed. What has been done as part of this project is essentially processing the data from one of these load steps. In general, the scanning x-ray diffraction analysis provides an abundance of data compared to just studying the relationship between the applied force and the compression of the grain collection.

Another experiment that might be considered is to measure continuously, or in small load intervals, while loading a single grain to the point of fracture. This analysis might provide insight in how the strains accumulate in the grain and if there are any characteristic behaviour before fracturing.

Of interest from a modeling point of view would be to compare the results here, where data have been recorded for the full volume of the grains with a finite element model. This might provide insights into which phenomena, e.g intra-granular friction or grain-wall contact, are necessary to accurately represent the system. Such a model is outside of the scope of the work done here, and will have to be considered in a different project. Another modeling project would be to find a reliable way to convert the measured strains to their corresponding tractions for the purpose of further understanding what happens at the points of contact.

The fact that the recorded diffraction peaks are unevenly distributed across the grains for large 2θ values remains a conundrum. Perhaps further information about the grains used, the quality of the crystals and more information about crystal structures in general can explain why the diffraction peaks are not distributed evenly across the grains.

Bibliography

[1] H. F. Poulsen, "3DXRD – a new probe for materials science", English, PhD thesis (2004).

[2] N. A. Henningsson, S. A. Hall, J. P. Wright, and J. Hektor, Journal of Applied Crystallography **53**, 314 (2020).

[3] N. A. Henningsson, *Scanning-xray-diffraction*, (2021) `https://github.com/AxelHenningsson/scanning-xray-diffraction`.

[4] J. Wright, *Imaged11*, (2021) `https://github.com/FABLE-3DXRD/ImageD11`.

[5] W. Borchardt-Ott, *Crystallography: An Introduction.* (Springer-Verlag Berlin Heidelberg, 2012).

[6] J. Als-Nielsen and D. McMorrow, *Elements of Modern X-Ray Physics* (Wiley, New York, 2000).

[7] E. Lauridsen, S. Schmidt, R. Suter, and H. F. Poulsen, Journal of Applied Crystallography - J APPL CRYST **34**, 744 (2001).

[8] H. F. Poulsen, S. F. Nielsen, E. M. Lauridsen, S. Schmidt, R. M. Suter, U. Lienert, L. Margulies, T. Lorentzen, and D. Juul Jensen, Journal of Applied Crystallography **34**, 751 (2001).

[9] A. Kak and M. Slaney, "3. Algorithms for Reconstruction with Nondiffracting Sources", in *Principles of Computerized Tomographic Imaging* (2001) Chap. 3, pp. 49–112.

[10] H. F. Poulsen and S. Schmidt, Journal of applied crystallography **36**, 319 (2003).

[11] A. Kak and M. Slaney, "2. Signal Processing Fundamentals", in *Principles of Computerized Tomographic Imaging* (2001) Chap. 2, pp. 5–47.

[12] M. Ristinmaa and N. Ottosen, *The Mechanics of Constitutive Modeling.* (Elsevier Science Limited, 2005).

[13] A. Borbely, L. Renversade, P. Kenesei, and J. Wright, Journal of Applied Crystallography **47**, 1042 (2014).

[14] U. Ayachit, *The ParaView Guide: A Parallel Visualization Application* (Kitware, 2015).

[15] P. Vestin, "Three-dimensional finite-element model of internal strains in a collection of silica grains", 2021.

Appendix A

Flowchart covering 3DXRD procedure and ASR algorithm.

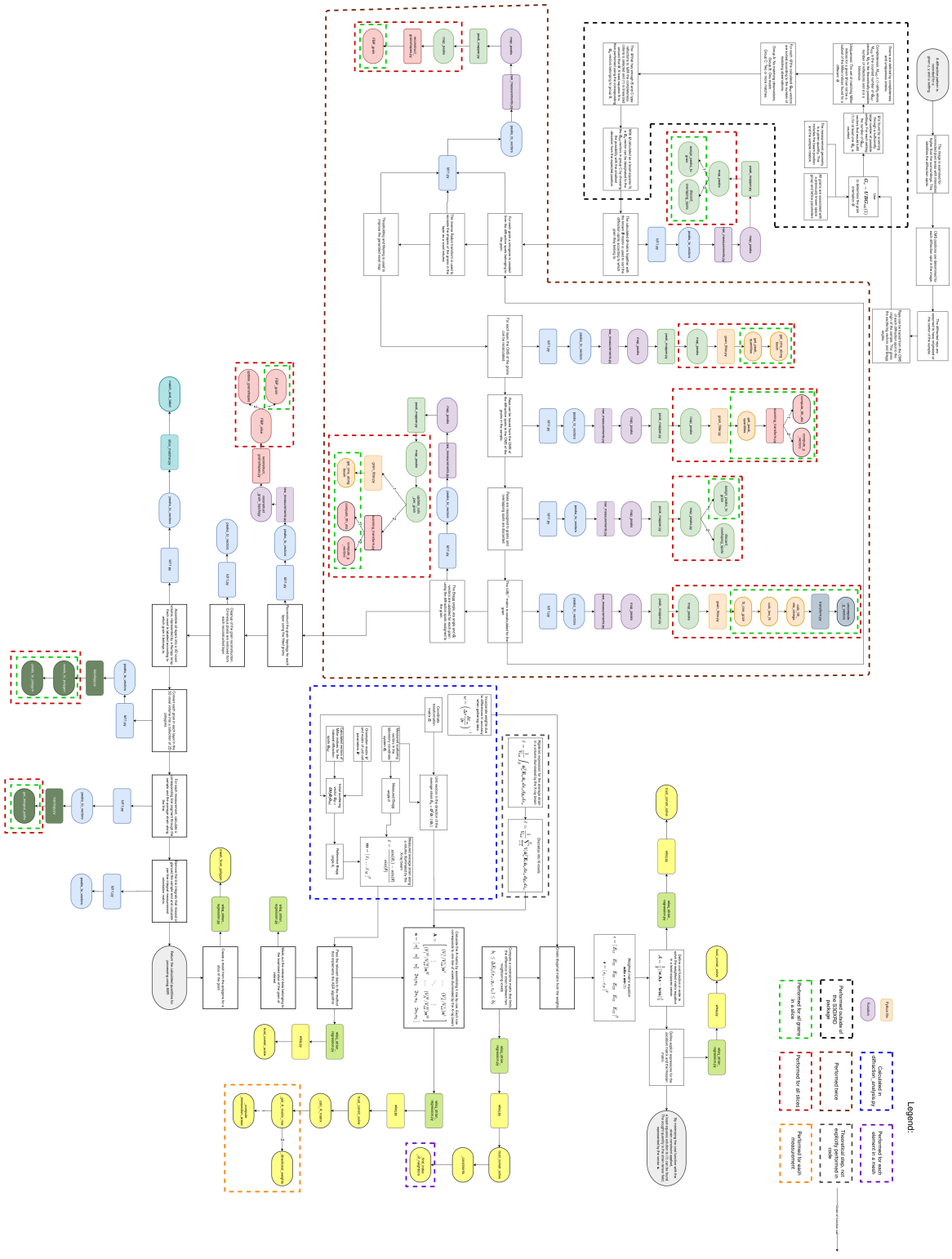


Figure A.1: Flowchart displaying the flow of data and the relevant code for processing recorded diffraction peaks and reconstructing the intra-granular strain fields.