FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

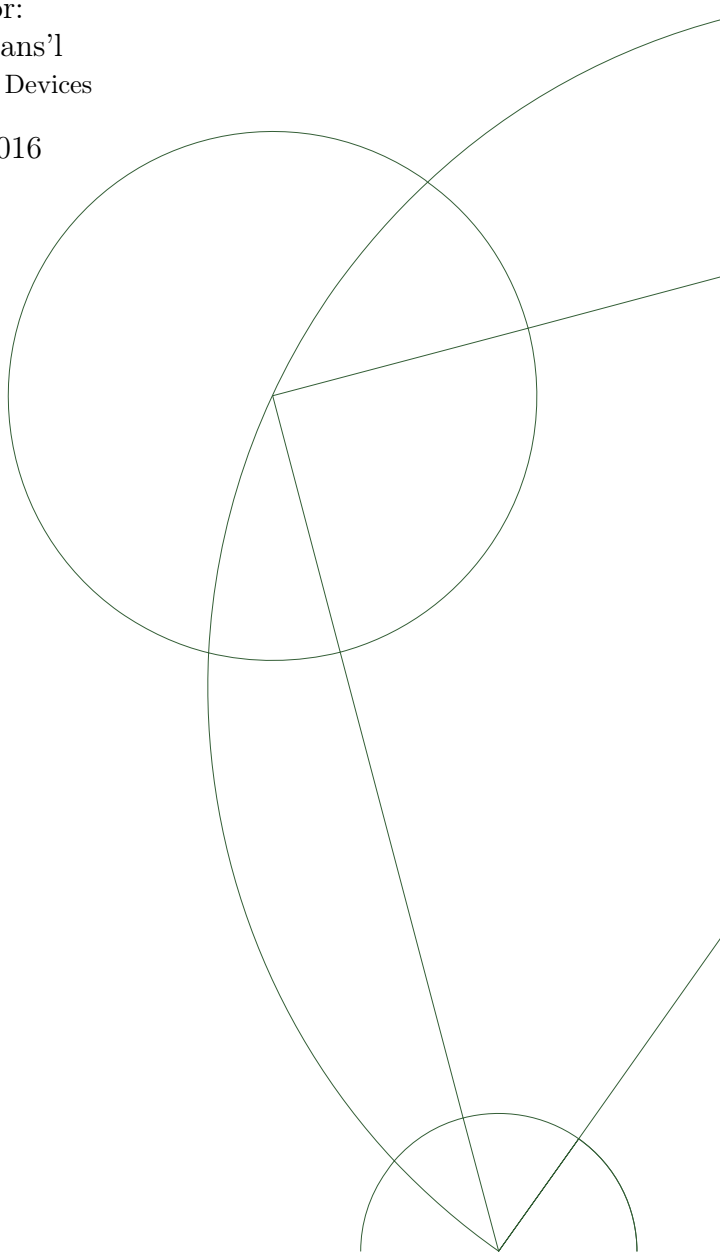
X-ray characterization of InAsSb nanowire heterostructures

Bachelor Thesis

Author: Rasmus Østrup Nielsen

Supervisor:
Peter Krogstrup
Co-supervisor:
Robert Feidenhans'l
Center for Quantum Devices

January 28, 2016



Abstract

Synthesis of nanoelectronics is a field still under development, leading to new discoveries and improvements of known technologies. This calls for structural characterizations techniques of different components, in order to find the relationship between structural and electronic properties. One widely used characterization method is X-ray diffraction which is a well known technique from crystallography. In this thesis, X-ray diffraction is described in order to conduct an experiment, aiming to characterize nanowires grown in the molecular beam epitaxy facility at the Center For Quantum Devices, under University of Copenhagen. The theory of nanowire synthesis and crystallography has been outlined, and the necessary theory of X-ray diffraction for detailed structural analysis has been described, in order to conduct and understand a scanning X-ray Diffraction Microscopy experiment and carry out the data analysis on the data obtained from this experiment.

In house grown nanowires were brought to the X-ray synchrotron facility PETRA-III at DESY in Hamburg. Several different heterostructural nanowires were brought to the experiment, and in this thesis we carry out detailed analysis of a sample consisting of $InAs_{1-x}Sb_x$/Al core/half shell. From this, the composition of the nanowires different regions was determined, and this showed that the incorporation rate of antimonide is higher than that of arsenide. It was further found that the strain induced by the aluminium half shell has a correlation with the antimonide concentration of the nanowire, with the strain growing larger with higher concentrations. This allows better strain engineering in future nanowire synthesis.

Contents

1 Introduction to nanowires

The worlds need for computational power is increasing, so miniaturization of transistors for processors have been of great interest since the first processor was constructed, and these transistors are now well in the nano-regime[4]. Now a new type of computation is under development, namely quantum computing. In order to make a working quantum computer, one of the promising approaches is utilizing Majorana states in nanowire-networks with high spin-orbit coupling[2]. In order to perfect the nanowire structures needed for quantum computing, it is necessary to have high resolution characterization techniques.

Nanowires are one dimensional crystals in terms of electron structure, and they are used in a wide range of applications. They have a large length to width ratio, and therefore resembles the wires we know from our daily life. The dimensions ranges from around 10 to hundreds of nanometres in width and micrometers in length. They are grown using the well-established crystal growth methods Molecular Beam Epitaxy (MBE) and Metalorganic Vapour Phase Epitaxy (MOVPE). In order to limit the growth to a single direction, giving the high aspect ratio, either catalysing nanoparticles or an inert mesh is applied to a substrate limiting the nucleation in width but not in height. The substrate consist of a flat crystal with well known crystal directions and with a crystal structure compatible with the structure of the desired wire. The wires are commonly made from III-V semiconductors, that is crystals of elements from the third- and fifth main group of the periodic table[1]. Different heterostructures can be made, either by differing the atomic composition along the growth direction, or when the desired length has been achieved, covering some of or all the sides with a different crystal (called axial- and radial heterostructures respectively). Since different crystals generally have different unit cells, such heterostructures requires lattice-matching, and for many geometries it would result in plastic strain relaxation being favourable over elastic strain, causing crystal defects in the wire. The high aspect ratio of nanowires however allows very effective elastic strain distribution, so that nanowires can be produced with a thick shell without causing dislocations in the interface. Strain calculations have shown that there exists a critical radius for which the shell can be made arbitrarily thick without causing dislocations, instead making perfect crystal matching along the interface. [7]

1.1 Crystallography

Crystals are by definition repeating structures in the sense they have translation invariance. In order to describe crystals, the mathematical concept of lattices is useful. A lattice is simply a regular array of points in space. A crystal is some unit cell which is repeated at every lattice point. These unit cells can be described by 3 vectors, $\mathbf{a_1}$, $\mathbf{a_2}$ and $\mathbf{a_3}$, called the lattice vectors. A crystal is thus a structure which is symmetrical under translation by vectors of the form

$$\mathbf{R} = n_1\mathbf{a_1} + n_2\mathbf{a_2} + n_3\mathbf{a_3}$$

where n_i is any integers. That means that an infinite crystal looks identical from all the vectors $\mathbf{R}$. Including other symmetries than this translational symmetry of $\mathbf{R}$, there is 14 different unit cells, ordered in 7 different types of coordinate systems called the lattice systems.[9] These are simply relations for the angles between - and lengths of the lattice vectors. Within these lattice systems there are 7 different ways the lattice points can be put. The simplest case is one lattice point at the end of each lattice vector, and other examples include the addition of a lattice point at each face or a lattice point in the middle of the unit cell. Taking into account other symmetries such as rotational, these can be divided into a larger number of crystals. Two crystal structures of particular interest for III-V compounds are wurtzite and zincblende, seen on Figure 1. Zincblende is a face centred cubic cell meaning that the lattice vectors $\mathbf{a_1}$, $\mathbf{a_2}$ and

[1] Using the old CAS-nomenclature. In the current IUPAC convention, it would be the 13th and 15th group. In short, III-V compounds are made from elements of the groups starting with Boron and Nitrogen

$\mathbf{a_3}$ has the same length and 90° between them and that there is an additional lattice point in the middle of each face of the unit cell. Wurtzite is a hexagonal unit cell. This means that the unit cell has a "bottom" plane, in which there conventionally lies 3 lattice vectors, $\mathbf{a_1}$, $\mathbf{a_2}$ and $\mathbf{a_3}$, of equal length and with 120° between them (Giving some redundancy, since $\mathbf{a_1} + \mathbf{a_2} = -\mathbf{a_3}$), and another lattice vector, $\mathbf{c}$, perpendicular to these 3 and generally of different length than $\mathbf{a_i}$, so wurtzite has 4 basis vectors. Zincblende and wurtzite are examples of direct lattices, meaning that they exist in real space. Both of these exhibits a hexagonal symmetry, which leads to layers along certain crystal directions, for wurtzite along the $\mathbf{c}$ axis (as introduced later, this direction is commonly called the [0001]-direction), and for zincblende along the $\mathbf{a_1} + \mathbf{a_2} + \mathbf{a_3}$ (Or [111]) direction. As seen in Figure 1, there are 2 layers in a wurtzite unit cell, leading to a ABABAB stacking sequence. Likewise, zincblende has 3 different layers in a unit cell, leading to ABCABC stacking. Heterostructures between these two are commonly produced, since they are very similar in structure along the axis perpendicular to the stacking layers, and thus are easy to make defect free, i.e. there are no dislocations (that is, broken bonds) in the interface between the two structures.

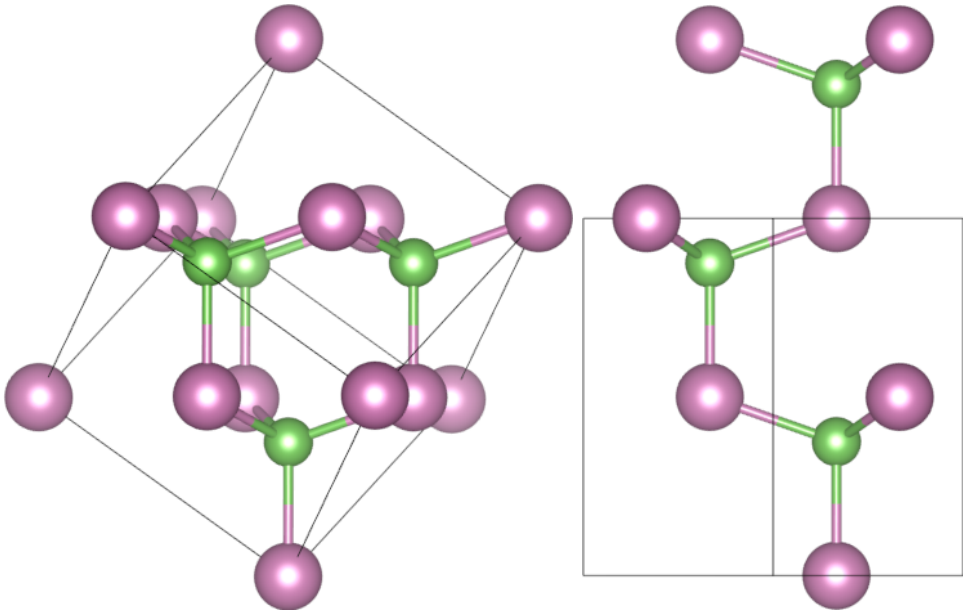


Figure 1: **Left:** The unit cell of the zincblende. The lattice-vectors are along the edges going out from the bottom atom. **Right:** The unit cell of wurtzite. The $\mathbf{a_i}$ lattice vectors are in the horizontal plane in which the bottom atom lies, and the $\mathbf{c}$ vector is vertical, extending from the bottom to the top of the drawn box.

A useful tool for analysing crystal structures is the reciprocal lattice. The reciprocal lattice is simply the Fourier transform of the electron distribution in the direct lattice. The electron distribution is periodic with the lattice, and this periodicity can be written as

$$n(\mathbf{r}) = n(\mathbf{R} + \mathbf{r})$$

where $\mathbf{R}$ is direct lattice vectors, and $n(\mathbf{r})$ describes the electron distribution in the unit cell. If we proceed by taking the Fourier transform, we get

$$n(\mathbf{R} + \mathbf{r}) = \sum_{\mathbf{Q}} n_{\mathbf{Q}} e^{i\mathbf{Q}\cdot\mathbf{r}} e^{i\mathbf{Q}\cdot\mathbf{R}} \tag{1.1}$$

Due to the translational symmetry of the direct lattice, the last exponential must be the same for all $\mathbf{R}$'s. Looking at the zero-vector $\mathbf{R} = \mathbf{0}$, this exponential must reduce to 1, so all other combinations of $\mathbf{Q} \cdot \mathbf{R}$ also have to reduce the exponential to 1. From this, we arrive at the

following equality:

$$\mathbf{Q} \cdot \mathbf{R} = N\pi \text{ For } N \in \mathbb{Z}$$

From this relationship, the reciprocal lattice can be constructed from all $\mathbf{Q}$ that satisfies this identity. This lattice also satisfies the translational symmetry, so it looks the same from every lattice point. It can be shown that the basis for the reciprocal lattice is given by [9]

$$\begin{aligned}\mathbf{b_1} &= 2\pi \frac{\mathbf{a_2} \times \mathbf{a_3}}{\mathbf{a_1} \cdot \mathbf{a_2} \times \mathbf{a_3}} \\ \mathbf{b_2} &= 2\pi \frac{\mathbf{a_3} \times \mathbf{a_1}}{\mathbf{a_1} \cdot \mathbf{a_2} \times \mathbf{a_3}} \\ \mathbf{b_3} &= 2\pi \frac{\mathbf{a_1} \times \mathbf{a_2}}{\mathbf{a_1} \cdot \mathbf{a_2} \times \mathbf{a_3}}\end{aligned}$$

From this it can easily be seen that $\mathbf{b_i} \cdot \mathbf{a_j} = 2\pi\delta_{ij}$

1.1.1 Crystal directions and planes: Miller indices

It is convenient to be able to characterize directions and planes in crystals in a simple way. This is done using Miller indices. If we have a reciprocal lattice with basis vectors $\mathbf{b_1}$, $\mathbf{b_2}$ and $\mathbf{b_3}$, then a direction in this lattice can be described by 3 integers, (hkl), which means the vector $\mathbf{Q} = h\mathbf{b_1} + k\mathbf{b_2} + l\mathbf{b_3}$. It can be shown that this point in reciprocal space is equal to the real space planes with surface normal $h\mathbf{a_1} + k\mathbf{a_2} + l\mathbf{a_3}$[9]. By convention, (hkl) denotes all these planes and $\{hkl\}$ denotes all planes which are symmetrical to this plane. Likewise, $[hkl]$ denotes the real space direction $h\mathbf{a_1} + k\mathbf{a_2} + l\mathbf{3}$, and $\langle hkl \rangle$ describes directions symmetrical to $[hkl]$. By convention, negative Miller indices are written with a bar, so the direction $2\mathbf{a_1} - \mathbf{a_2}$ is written $[2\bar{1}0]$. Since, as mentioned above, wurtzite has 4 basis vectors, there are 4 Miller indices, $(hkil)$, to describe planes and directions in that system. The 3 first, h, k and i are linearly dependant so $h + k = -i$.

1.1.2 Axial heterostructure: Compositional variation

When growing axial heterostructures, the aim can be to introduces a material which is hard to grow to a material which is easier grown. For example, InSb shows promising superconducting properties, but is hard to crystallize in a controlled manner[3]. One method to overcome this is to start by growing a short segment of InAs wurtzite, and then introduce a flux of antimonide as well. Since the InSb unit cell is larger than InAs, the second segment must have a low antimonide concentration to match the interface. If higher antimonide concentrations are required, multiple segments are grown with the concentration increasing for each segment. This is because if the lattice mismatch between two segments is too large, the strain induced will make the interface incoherent, such that not all atoms are able to form a bond, thus leading to defects in the interface. This is known as plastic strain relaxation. If the crystals are well matched, we will have an elastic strain, meaning that the unit cell is stretched or compressed in order to compensate for the lattice mismatch.

It is convenient to have a relationship between the lattice parameters of the two structures going into an alloy, and thus a heuristic relationship called Vegards Law is introduced. It has no theoretical justification, but since there has not been found notable deviations from it, it is still the way to calculate lattice parameters of alloyed compounds. For $InAs_{1-x}Sb_x$, it is given by

$$a_{\mathrm{InAs}_{1-x}\mathrm{Sb}_x} = 1 - x \cdot a_{InAs} + x \cdot a_{InSb}$$

This off course, can be inverted, so if one performs a measurement on the lattice constant, it can be converted back to a concentration.

1.2 Strain Mapping

In recent years electronics have been further miniaturized. This has led to the development of techniques for characterizing the components in very high resolution. One of the characteristics that has been found to be of importance for the performance of devices is the internal strain of nano scale crystal structures[10]. Strain is essentially the deviation from the theoretical perfect unit cell of the crystal in question. The definition of strain

$$\epsilon_{xx} = \frac{a - a_0}{a_0} \tag{1.2}$$

is the ratio between the deviation from the perfect unit cell and the perfect unit cell, and is thus giving how much the unit cell has expanded or contracted in the given direction. In order to map these, different crystallographic methods have been applied and perfected. These are mainly Transmission Electron Microscopy (TEM), Micro-Raman Spectroscopy (μ-RS) and X-ray diffraction (XRD)[5]. X-ray diffraction is limited by the ability to focus the X-rays, but methods like convolution has been applied to improve the resolution. The convolution however, does not improve the resolution beyond that of TEM, but TEM is only usable for samples less than a few hundred nanometres thick with uniform strain distribution in the depth. Making a full strain map using TEM of a structure bigger than this requires the sample to be sliced into bits, small enough that the TEM can probe through these. This destructive sample preparation may change the strain in the structure, so even though the resolution is higher for TEM than for XRD, XRD still has the advantage of a better probing depth. Raman Spectroscopy is an indirect method where photons interact with the vibrational modes of the crystal lattice. This method is limited by the ability to focus the light source, and the best methods can provide a resolution of about 0,3µm. As this is the width of a large nanowire, this method is not very suitable for measuring strain variations within nanowires.

1.3 X-Ray Diffraction

As discussed above, XRD is a well suited method for measuring strain in nanowires. The method makes use of the regular distribution of the electrons within the crystal lattice and photons ability to scatter elastically when interacting with these. When a photon interacts with an electron, a secondary scattered wave arises, and propagates circularly out from the scatterer electron. Since the scattering is elastic, only the direction of the wave changes, but the energy remains the same. When using X-Ray diffraction methods, the incoming X-rays are monochromatic, and hits a large portion of the crystal, making a regular array of circular waves from the different lattice points. These will cause destructive interference in most directions, but constructive interference can occur at some angles of incidence relative to the crystal lattice. These are given by the Bragg-condition[9]:

$$n\lambda = 2d_{hkl}\sin(\theta) \tag{1.3}$$

Where d_{hkl} denotes the distance between the (hkl)-planes in real space. In the case of wurtzite, these will be replaced with d_{hkil}.

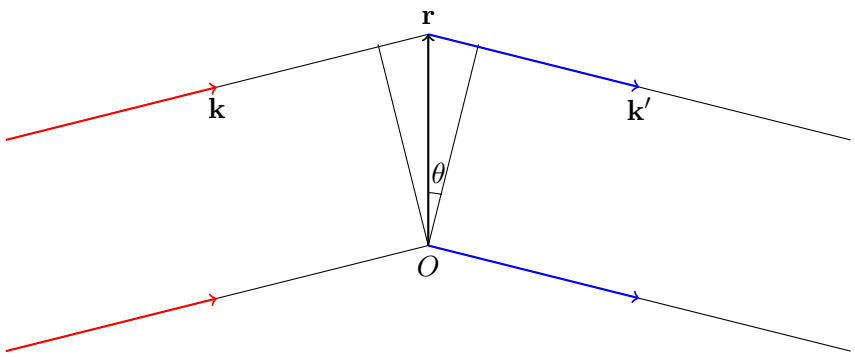


Figure 2: Sketch of incoming (red vectors) and outcoming (blue vectors) waves scattering from a crystal

The Bragg condition gives us a way to measure the distance between crystallographic planes. It can further be shown that the reciprocal lattice of the crystal directly gives the possible diffractions from a given crystal. In order to see this, we must define a scattering amplitude. A small volume element dV at $\mathbf{r}$, has the scattering electron density $n(\mathbf{r})$. The incoming beam of X-rays have the wave vector $\mathbf{k}$ and the scattered wave has $\mathbf{k}'$. The phase difference of the incoming wave at points O and $\mathbf{r}$ is the distance the angle of incidence spans as seen on Figure 2. This is given by $\mathbf{k}\cdot\mathbf{r}$. Similarly the phase difference at the outgoing beam is $-\mathbf{k}'\cdot\mathbf{r}$. This means that phase difference between the outgoing wave vectors from O and $\mathbf{r}$ is given by the phase factor $e^{i(\mathbf{k}-\mathbf{k}')\cdot\mathbf{r}}$. Integrating over the whole crystals electron distribution multiplied by the phase difference between $\mathbf{r}$ and O of the scattered waves gives the scattering amplitude[9]:

$$F = \int n(\mathbf{r}) e^{-i\boldsymbol{\Delta k}\cdot\mathbf{r}} dV$$

where $\boldsymbol{\Delta k} = \mathbf{k}' - \mathbf{k}$. Combining this with the Fourier transform of the electron distribution obtained in equation 1.1, we get

$$F = \sum_{\mathbf{Q}} \int n_{\mathbf{Q}} e^{i(\mathbf{Q}-\boldsymbol{\Delta k})\cdot\mathbf{r}} dV$$

When $\boldsymbol{\Delta k} = \mathbf{Q} = m_1\mathbf{b_1} + m_2\mathbf{b_2} + m_3\mathbf{b_3}$, the scattering amplitude is maximized to the value $F = Vn_{\mathbf{Q}}$, and vanishing quickly when $\boldsymbol{\Delta k} \neq \mathbf{Q}$. If we take the scalar product of this condition with the 3 direct lattice vectors, we arrive at the following relationship known as the Laue conditions[9]:

$$\begin{aligned}\mathbf{a_1}\cdot\boldsymbol{\Delta k} &= 2\pi m_1\\ \mathbf{a_2}\cdot\boldsymbol{\Delta k} &= 2\pi m_2\\ \mathbf{a_3}\cdot\boldsymbol{\Delta k} &= 2\pi m_3\end{aligned}$$

These equations have a significant geometric interpretation, namely that each of them specify a cone where diffraction is possible. Since all of these three equations must be satisfied, it means that reflections can only occur at the intersect of the three cones. This relationship is beautifully visualized by the Ewald Sphere, seen on Figure 3

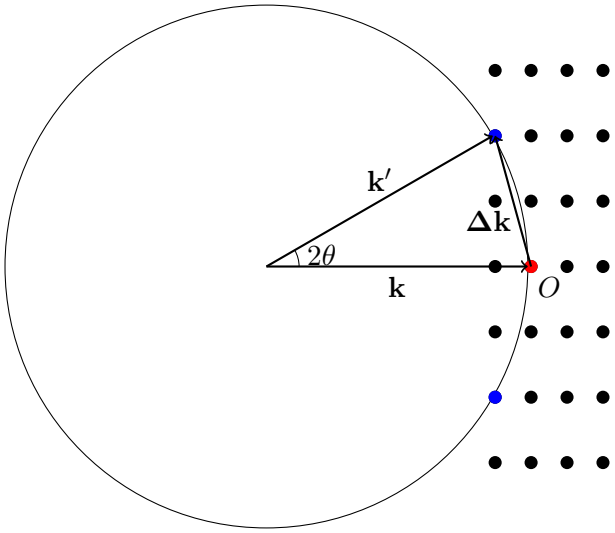


Figure 3: The Ewald sphere. O, the red point, is any point in the reciprocal lattice and the incoming wave vector is terminating at this. The blue points in the reciprocal lattice are the ones which can diffract for the given angle of incidence and incoming wave vector length.

Due to the elastic scattering, the incoming and outgoing wave vectors has the same length, so if they start from the same point, they will both end on a circle of radius $|\mathbf{k}| = |\mathbf{k}'| = \frac{2\pi}{\lambda}$. The incoming wave vector is set to point at any reciprocal lattice point. The Ewald sphere then shows how diffraction will form if the sphere intersects any other reciprocal lattice point.

1.3.1 In-plane and out-of-plane diffractions

It is useful to distinguish between in-plane and out-of-plane diffractions. These terms are relating to the substrate orientation, so diffraction signals which comes from planes parallel to the substrate surface normal are in-plane peaks because the substrate, and the wave vectors will be in parallel planes. This causes the diffraction signal to be in the same plane as the substrate for the perfect unstrained lattice. In the case of the experiment we describe later, the substrate is an InAs zincblende substrate, with [111] as the surface normal. A wurtzite-zincblende nanowire heterostructure is grown on this substrate. The [111]-zincblende direction is parallel to the [0001] direction of wurtzite. Now that we have the relevant surface normals defined, we can find the planes which are parallel to these by taking the scalar product with the surface normal and the diffraction plane normal. If this is 0, we have an in plane peak. That gives these requirements: For wurtzite the in-plane peaks are coming from $(hki0)$-planes, and for zincblende they are coming from planes (hkl), where $h + k + l = 0$. The geometry of these planes and their diffraction signals makes it particularly easy to separate the d_{hkl} values from the angle the plane is tilting. The planar spacing is given by the in-plane reciprocal distance, while the tilt is the out-of-plane reciprocal distance.

2 Scanning X-ray Diffraction Microscopy

Scanning X-ray Diffraction Microscopy (SXDR) is a method for making highly detailed reciprocal space maps of nanowires. It utilizes the theory of X-ray diffraction, but instead of illuminating the whole sample at once, a series of diffractions are measured on a grid on the sample by moving the wire around in the highly focused X-ray beam, in our experiment as small as 73nm full width at half maximum. This gives an array of Bragg-peaks from different points on the sample, meaning that the reciprocal lattice vector for these points can be calculated, as sketched on Figure 4. These reciprocal lattice vectors translates into a plane spacing in real space, which is a

measure of the lattice constant. To perform a SXRD-experiment, a pixel-detector is placed on the Ewald Sphere at the position of a chosen Bragg-peak, and the sample is rotated along ω to find the orientation corresponding to that peak. For measurements on axial heterostructures with a gradient composition, the reciprocal lattice points will move slightly when the composition changes, i.e. each concentration has different planar spacing. This means that the points of diffraction will change slightly. In order to compensate for these changes, one can rotate the sample a small ω around the nanowire-axis and perform another SXRD. Since the reciprocal lattice rotates with the sample, different parts of reciprocal space will be on the detector at different sample rotations, and thus one can get a measurement of a whole sample, even if the composition changes, and thus drifts in and out of Bragg condition on the detector. Then it is a matter of piecing the detector images from the SXRDs with varying sample rotation together. This will form a 3D Bragg peak, from which the structural characteristics of the sample can be calculated.

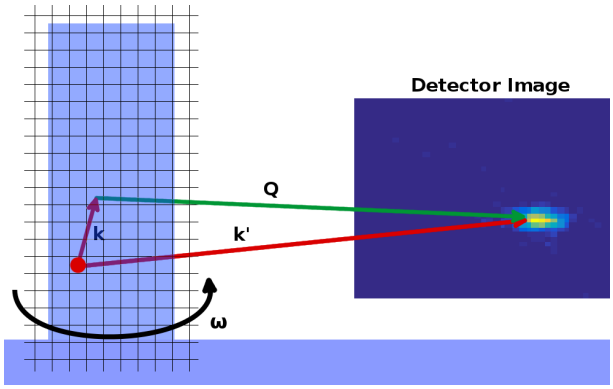


Figure 4: A simple sketch of SXDR. At the bottom is the substrate, and the nanowire is standing on the left side of it. On the nanowire, the scanning grid is drawn, and the red dot represents the incoming X-ray beam at the point being scanned. To the right is the detector image of the point being scanned. The incoming wave vector is really perpendicular to the scanning grid plane, and of the same length as the diffracted wave vector due to energy conservation. The difference between them, **Q** is sketched as well, showing how the plane spacing is measured for each scanning point. The angle ω is the sample rotation around the nanowire preserving the nanowire position and surface normal of the substrate.

3 Experimental setup

To characterize strain and composition of a nanowire in great detail, a series of scanning X-ray diffraction microscopy analysis was carried out at Deutsches Elektronen-Synchrotron (DESY), at their PETRA-III synchrotron. The experiments were carried out in the P06-beamline which have nano-focusing lenses that can focus the monochromatic X-ray beam down to around 80nm. This makes it ideal for SXRD. A number of samples were brought along, all with the common feature of being core/(half) shell structures. The aim was to measure the strain in aluminium shells of the different wires and using convolution to improve the resolution of the measurements to less than the beam size. Convolution is a method where the difference is taken between overlapping scan points. Then, using the difference between these overlapping measurements as the new data points, the resolution can be improved a lot. Due to drifts in the system, this was not possible to achieve since the overlap of the different scan points had a large uncertainty to them, meaning that we could not tell how much the points are actually overlapping. Unfortunately, we neither found any aluminium Bragg-peaks, and had to abandon this as well. Instead, we ended up with lots of measurements on the cores of the wires, and as we shall see later, these measurements for the analysed sample, shows signs of the aluminium shell inducing strain in the NWs.

3.1 Sample overview and preparation procedure

The focus of this thesis is on a particular sample, which is both an axial- and radial heterostructure. It is seen sketched on Figure 5. It was grown in a MBE, on a InAs substrate in a zincblende

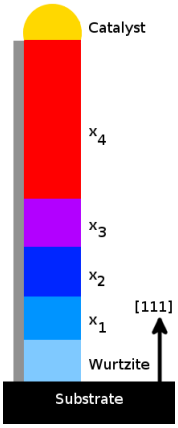


Figure 5: Overview of the analysed nanowire. The different segments are color-coded, and the axial parts are labelled. The grey segment on the left side is the aluminium half shell.

lattice with the [111] direction perpendicular to the surface. The core was grown in 5 sections. First an InAs wurtzite-stem with the [0001] direction parallel to the [111]-zincblende direction. On top of the wurtzite stem is 4 zincblende-sections of $InAs_{(x-1)}Sb_x$. The flux-values from the MBE for the four sections are $x_{v,1} = 0.15$, $x_{v,2} = 0.2$, $x_{v,3} = 0.26$ and $x_{v,4} = 0.55$. Further, an aluminium shell is on 2-3 of the nanowire facets, causing the wires to bend slightly. The wires are about 110nm wide with 10nm being the Al-shell and $3 - 4$µm high including the InAs wurtzite stem. A SEM-picture of the sample is seen in Figure 6. In order to map a single wire using SXRD, some space around the wire is necessary. This is to make space for the X-ray beam hitting a single wire, so a diffraction signal is measured from just this one nanowire. Since zincblende and wurtzite have a 6 fold symmetry around the [111] and [0001] direction respectively, a Bragg peak will repeat every 60° of sample rotation ω. This also means that in order to be able to find the Bragg peak when the detector is put on the Ewald Sphere at the predicted angle of a Bragg-peak, there has to be room for 60° rotation along ω for a single nanowire to ensure that it can be rotated into Bragg condition without other wires interfering. In order to achieve this, the sample was prepared such that wires was standing on a 60° substrate corner, and along one of the edges. Behind this line of wires, all other wires were removed for a couple of microns. This gives 120° of free space around the wire furthest out on the substrate tip. This is seen on Figure 6.

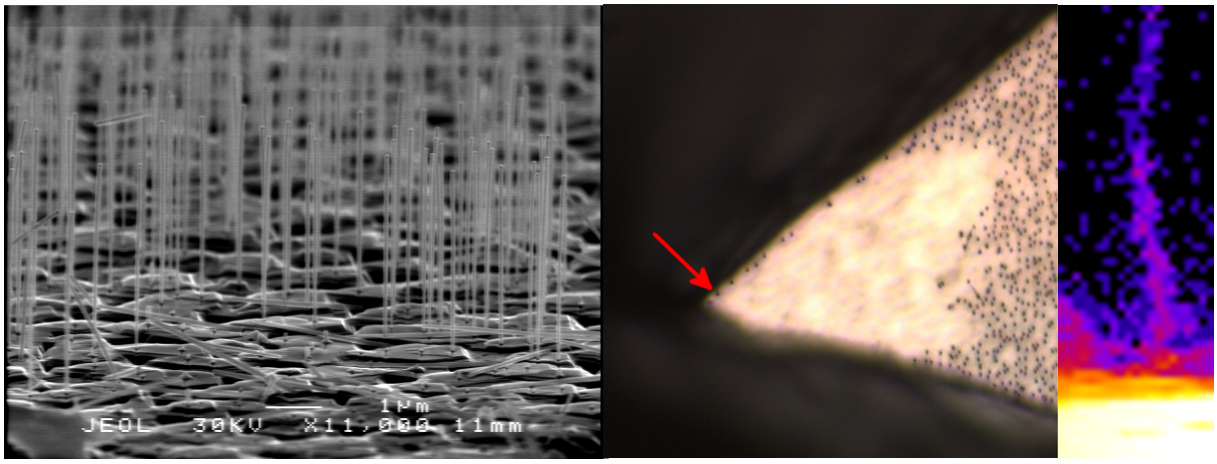


Figure 6: **Left:** SEM picture of the sample analysed. **Middle:** Microscopy picture of prepared sample. The red arrow points to the nanowire measured. **Right:** Fluorescence map of a wire. Comparing with the SEM picture, the drift in the laboratory system is clearly seen.

3.2 Beamtime

The layout of the detectors and scanner units of the P06 beamline at PETRA-III are seen on Figure 7. The prepared sample were mounted on a sample holder with carbon tape, which was

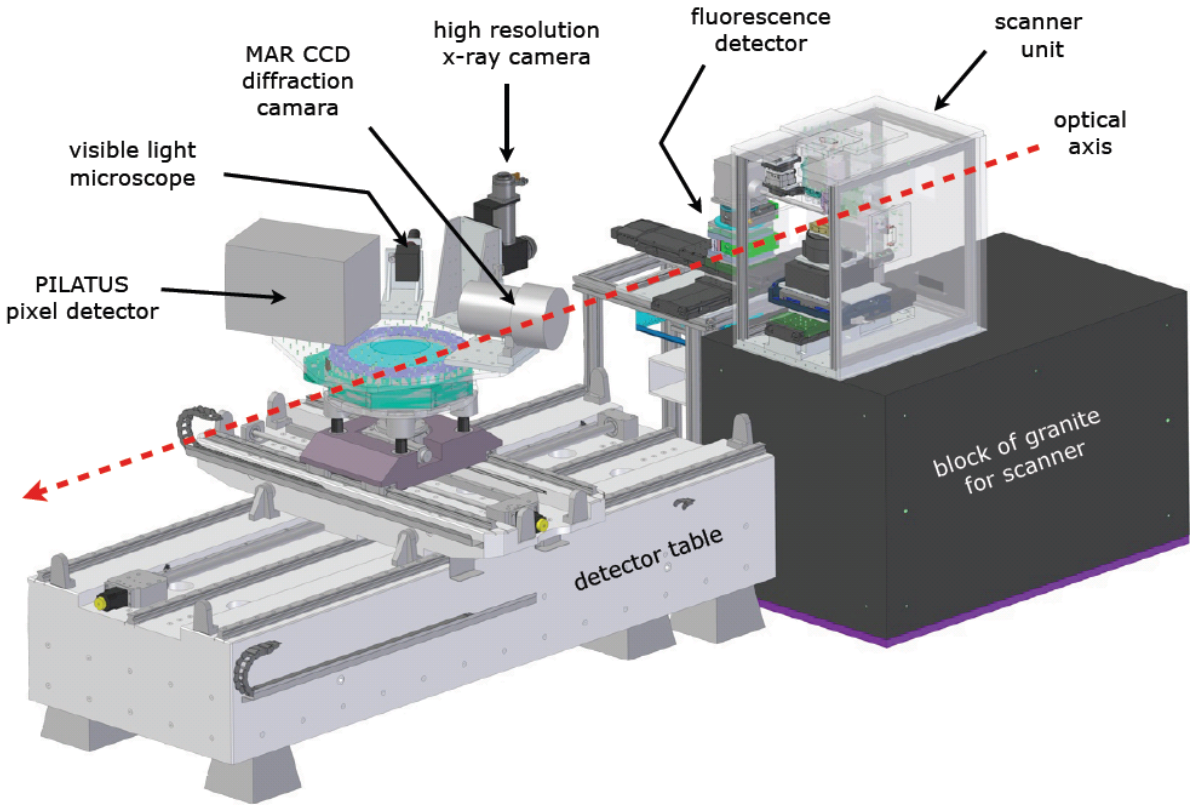


Figure 7: Layout of the detectors and scanner unit. Inside the scannerunit, the sample is mounted on a sample stage, and located on the optical axis. The interferometers are opposite the fluorescence detectors. The CCD and X-ray camera were not used in this experiment. (Courtesy of DESY[1])

then mounted on a sample stage (located inside the scanner unit) with the substrate horizontal. The sample stage is capable of moving the sample around with very high precision, and has both scanning motors for moving the sample around in a scanning grid and adjustment motors for

placing the sample at the desired position on the optical axis. To locate the wires on the sample, an optical microscope was used. This microscope was set up, such that its focus at full zoom was in the same plane as the X-ray beam focus. This microscope was also used to position the wire in the centre of rotation of the sample stage, such that the sample could be rotated while the wire to be analysed stays in the X-ray focus. Bringing the wire to the centre of rotation of the sample stage was done by locating a wire, rotating the sample stage 180°, then moving the wire to halfway between the two positions it was observed at. For a finer alignment, X-rays were engaged and the wires were located using fluorescence measurements. The process of bringing the wire to the centre of rotation was repeated with the fluorescence detector to fine tune the wire position to the centre of rotation. During the alignment process, it was noticed that the sample was drifting a lot, giving rise to snake-like wires. This is seen in Figure 6. Throughout the experiment, the fluorescence detector was mapping the wire position and due to the drift, this became of importance, since the sample stage had to be adjusted if the wire drifted out of the scanning area. Further, the sample position was measured using interferometry, to have data on drift during scans, in order to straighten out the maps of the wires to their original shape. When the wire was in the centre of rotation, a wide rocking curve with 120° span, and with detector close in order to cover a wider angular range of the Ewald Sphere, was measured. From this, 4 different peaks were found at 4 different angles. Then, calculating the angles between the direct beam and the peak, we determined that it was, with respect to the wurtzite lattice, the $(2\bar{1}\bar{1}0)$, $(11\bar{2}\bar{1})$, $(10\bar{1}1)$ and $(1\bar{2}10)$. From this it was chosen to work on with the $(2\bar{1}\bar{1}0)$ peak for various reasons. The position of the $(2\bar{1}\bar{1}0)$ InAs wurtzite peak or equivalently the $(3\bar{3}0)$ (Or the third order diffraction of the $(1\bar{1}0)$) zincblende peak was calculated using the Bragg condition combined with the fact that the measured plane is perpendicular to the substrate, leading to only a horizontal displacement of the detector. These peaks are at the same position in reciprocal space, and they are in-plane peaks, meaning that the reflecting plane is parallel to the surface normal of the substrate, so that both the incoming beam and the diffracted beam will be in a plane parallel to the substrate in the ideal unstrained crystal. This is convenient because the distance to the Bragg peak projected into this plane will correspond to a planar spacing, while an offset from the plane will mean that the reflecting plane is not exactly parallel to the substrate surface normal, which corresponds to how much the wire is bent. The Bragg-angle for the $(2\bar{1}\bar{1}0)/(3\bar{3}0)$ peak is $2\theta \approx 30°$, and lies in the same plane as the substrate, so the pixel detector was placed accordingly. To improve the angular resolution of the peak, the detector was brought back, so each pixel spanned a smaller angle. A finer rocking curve of $\omega = \pm 1°$ was then performed, to see at which sample rotation angles, ω, we could expect intensity over the background level. This curve is seen in Figure 8. Covering the ω range of high intensity in this curve, different scans of 20 points over 400nm in width and 4µm over 70 points in height were done. This gives 20nm between each column, and 57nm between each row. This is smaller than the beam size of 73nm, so there is some overlap between scanning points. This was done in order to improve the resolution beyond that of the beam size, by using convolution. As explained, this was abandoned.

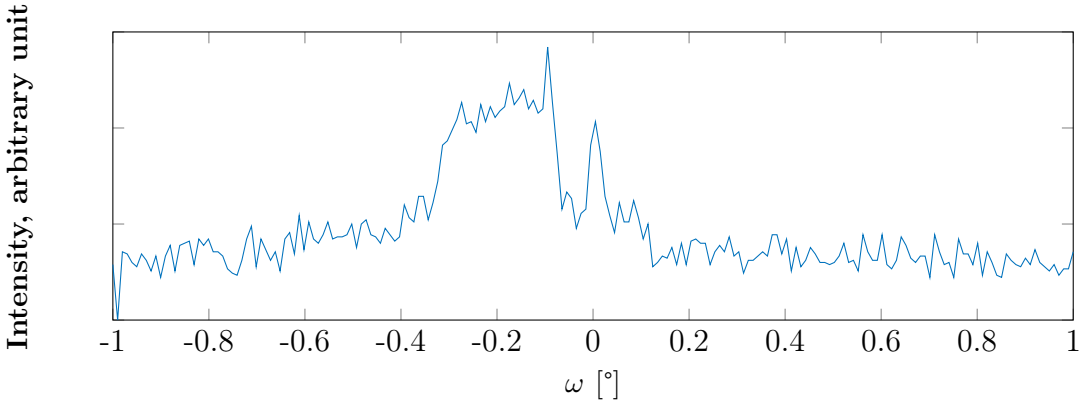


Figure 8: Rocking curve around the $(2\bar{1}\bar{1}0)/(3\bar{3}0)$ peak.

3.3 Data

In order to get an overview of the data obtained, the total Bragg peak of all the scans are plotted. This was done in the following way: For each different ω, all the detector images from the different points in the scanning grid were averaged to obtain a single image. The average intensities were then filtered by leaving out values under a certain threshold, so the features we are trying to see, do not drown in background signal. Then the reciprocal space positions of each detector-pixel was calculated. These are different for each ω, since the wire, and therefore reciprocal space has been rotated. By plotting the filtered intensities at their corresponding positions, we arrive at the Bragg peak seen in Figure 9. The axes are chosen such that Q_z is parallel to the wire, and Q_x and Q_y are parallel with the substrate, so that the direction corresponding to tilt and strain are easily separated. In this space, the predicted position of the $(2\bar{1}\bar{1}0)/(1\bar{1}0)$ peaks for pure InAs is $(1.5, -\frac{\sqrt{2}}{2}, 0) \approx (1.5, -0.866, 0)$. Deviations from this is caused either by the sample not being mounted perfectly vertical, or by lattice constant differences in the nanowire. The sample not being mounted perfect will not cause problems in the further analysis, since the scalar deviation from the theoretical positions will apply to the calculated quantities as well. Therefore, it is a matter of setting the reference point for the unstrained peak at an unstrained part of the wire, for which the substrate peak is the right candidate. By using a program that can correlate the intensity data to the part of the wire which the intensity came from, it was discovered that this is overlapped with the wurtzite stem peak. This is within expectations since we are measuring an in-plane peak, and wurtizite and zincblende has the same in plane lattice constants, meaning that the lengths of the zincblende lattices vectors projected onto the substrate plane has the same length as the 3 in-plane wurtzite lattice vectors. The wurtzite and substrate peak is the gathering of data points on the lower right of the full Bragg-peak. Above that, a peak from substrate growth was found, and on a downwards slanted line from this, 3 relatively well separated peaks are seen. These are from the 3 lower parts of the InAsSb-alloy, with the one furthest from origo corresponding to the lowest part in the wire. This fits well with expectations, since a higher reciprocal distance means a lower planar spacing. To the left, the x_4 peak coming from the top part of the wire is found. This has the highest antimonide concentration, and therefore is closest to the origin. This is a peak is very elongated in the Q_z- (out-of-plane) direction, so here we see a hint that the high concentration of antimonide is leading to a higher tilt of the reflecting planes.

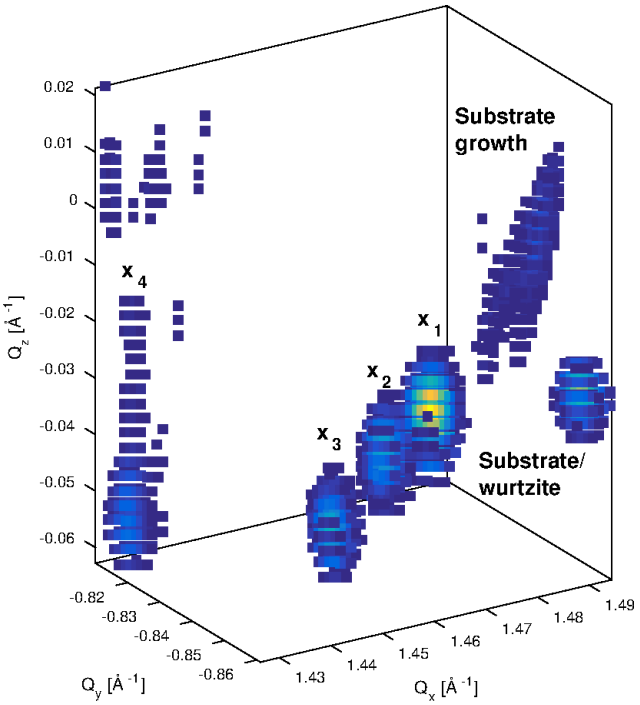


Figure 9: 3D Bragg peak, filtered so low intensities does not block the view. The peaks comming from different parts of the wire is labelled

Due to the drift in the system, every scan was treated separately in order to be able to correct them with the interferometer measurements. Before proceeding to this the intensity threshold was lowered compared to the ones plotted in the 3D Bragg-peak. This was to ensure that all diffraction signals were included. For every scan point in every scan, the center of mass (COM), that is the average of the detector-pixel position weighted by the corresponding intensity, was calculated in this way:

$$COM_x = \frac{x_i \cdot I_i}{\sum I_i}$$

Where the i-index denotes different detector-pixels. Doing this for all three directions, and for every scan point, 3 arrays containing the COMs of each real space point was created. For the further analysis, a fourth array was defined as in-plane COM:

$$COM_{hor} = \sqrt{COM_x^2 + COM_y^2}$$

We also need an unstrained reference point in order to calculate composition and strain. Therefore, the 3D-Bragg peak was cropped down to only contain the wurtzite/substrate peak, and the COM was calculated for this. This was done by taking the average of each scan COMs weighted by their intensity in the region of interest. The value of this came to

$$(COM_{x,ref}, COM_{y,ref}, COM_{z,ref}) = (1.491\text{Å}^{-1}, -0.8574\text{Å}^{-1}, -0.03252\text{Å}^{-1})$$

Due to different parts of wires having different intensities at different ω, an array of the sum of intensities from the detector pixels for each point in the scanning grid were saved for each scan as well, for weighting the data from different parts of the wire.
The COM- and intensity arrays were then drift corrected by the 3 interferometers. These were at angles 0°, 30° and 60° relative to the substrate plane. So by doing the trigonometry for the situation, splitting the 30° and 60° measurements into horizontal- and vertical drift, then averaging the coordinate shifts for the 3 interferometers, every scan point was moved to the

correct position. This caused some of the positions in the array to contain no data, so an interpolant were calculated for the corrected scan-points, and from this interpolant, interpolated arrays were calculated to fill in the positions that was moved by the interferometer corrections. In order to calculate the lattice constant at each scanning point, first the horizontal distance to the reference peak was calculated to $COM_{hor,ref} = 1.7199\text{Å}^{-1}$. The ratio between the measured lattice constant and the reference lattice constant $\frac{a}{a_0}$ can easily be converted to reciprocal lattice constant. By definition $a = \frac{1}{Q}$, so $\frac{a}{a_0} = \frac{Q_0}{Q}$. Then, multiplying by the theoretical lattice constant for InAs zincblende[2] $a_0 = 6.0583\text{Å}$, the measured lattice constant can be calculated as $a = \frac{Q_0}{Q} \cdot a_0$. This was done for all elements in the COM_{hor}-arrays, so we get an array of lattice constants for each scan. Vegards law can be inverted to

$$x = \frac{a_{InAs_{1-x}Sb_x} - a_{InAs}}{a_{InSb} - a_{InAs}}$$

The value of $a_{InSb} = 6.479\text{Å}$. From these, we can calculate arrays with the composition values for all scan points in all the scans. Now we have 3 types of arrays: Lattice constant arrays, composition arrays and intensity arrays. There are 6 of each, one for each scan. These are all corrected for the drift in the system in their internal coordinate system, but not corrected to an overall system. In order to visually align these, the lattice parameters at different ω were plotted side by side. From these plots it was seen that the horizontal drift averaged out during the scan, such that the NW where seen at the same horizontal scan points. The vertical drift seemed to have a tendency to drift downwards between the scans, so from the first to the last scan, the substrate had drifted out of the picture. To correct these drifts, arrays with the original scan width (20 points) and height of the sum of the greatest offset between a reference point in the different scans and the original height of 70 points. The 3 sets of original arrays were then put into these at the appropriate heights. Then it was checked that the interfaces between segments were aligned for the different scans, and it was confirmed that these all lined up after this procedure. Further, to give the different scans the same weight the intensity arrays were normalized so their elements sum to 1, and further, all the rows of scan points was normalized as well to give the same intensity at different heights of the wire. Due to the different Bragg conditions and the filtering of low intensity-values, after the row normalization there was still some rows of nearly 0 intensity corresponding to the parts of the wire which for the given scan was out of Bragg condition. These almost 0 intensities where then set to not a number (NaN) in the array, and the intensity where then averaged ignoring the NaNs, so that the low intensity values would not lower the intensities on the other scans. The other two final arrays of lattice parameters and composition where then calculated like this:

$$a_{x,y} = \frac{\sum a_{x_j,y_j} \cdot I_{x_j,y_j}}{6 \cdot I_{avg,x,y}}$$

Where the sum goes over j which denotes the different scans. x and y denotes the scanning points. I is the intensity in the 6 individual scans and I_{avg} is the average intensity over all the scans after they have been aligned properly. The 6 is there because there is 6 scans, otherwise the values would have been 6 times too big. The composition x was calculated in the same way, substituting a with x.

4 Results

From the data processing described above, we arrive at the picture seen in Figure 10. Starting from the bottom and going upwards, we see that the substrate has vanished. This is probably

[2]Since our reference is in unstrained InAs zincblende

due to the substrate having drifted out of the scanning area in most of the scans, so averaging the intensity makes the substrate intensity almost 0. Above the missing substrate, we see 2 layers of substrate growth with antimonide concentrations around $x = 0.2$ and $x = 0.4$, roughly corresponding to the 1st and 3rd InAsSb-sections. Another local study of InAsSb wires, showed that more antimonide would end up in the substrate growth than in the wire itself. These values might be due to that effect. However, since this wire has multiple segments of different concentrations, it is hard to analyse this relationship, since it is hard to distinguish the different parts of the substrate growth.

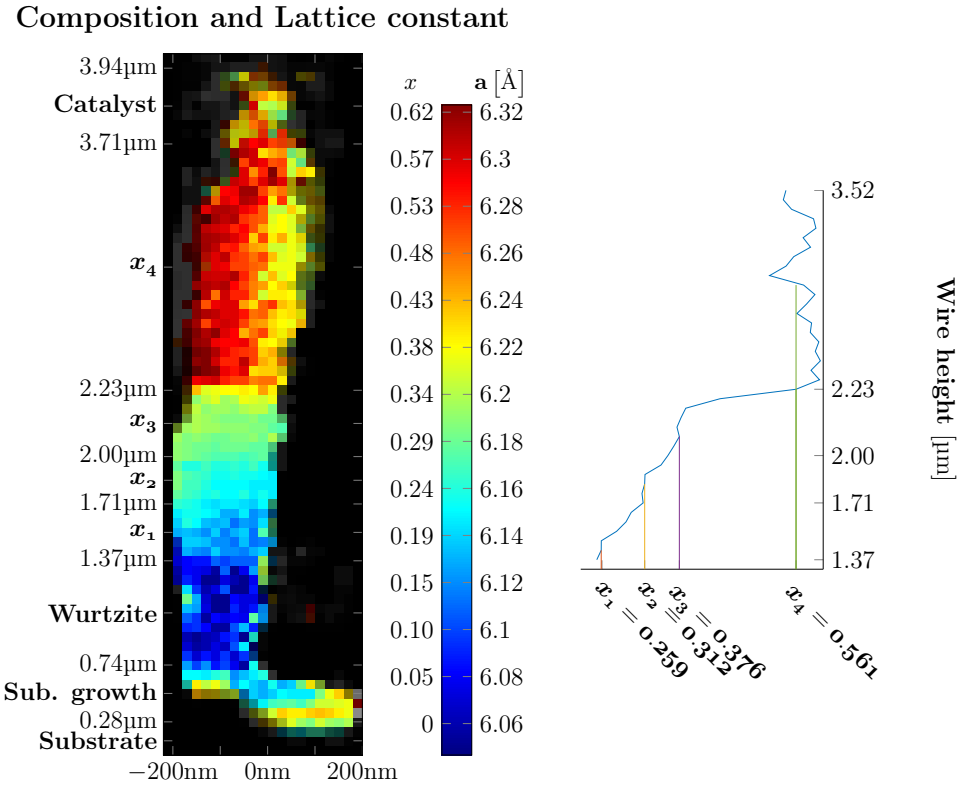


Figure 10: **Left:** A plot of the lattice constants and concentration values for antimonide in the wire. **Right:** The wire height as a function of the mean of antimonide concentrations. It is aligned with the 2 other plots, such that the constant x-values corresponds to the height on the real space NW pictures.

Going up from the substrate growth, we see the wurtzite stem. Due to the different lattices of wurtzite and zincblende, the lattice constant is of, because it is calculated as zincblende. Instead it is at the value of InAs zincblende. This is expected since the zincblende axis projected onto the substrate plane has the same length as the wurtzite in-plane lattice vectors. We also see that the antimonide concentration in this part is around 0. This is again caused by wurtzite and zincblende having the same in-plane lattice parameters, so this is a reliable measurement and as expected since when this part was synthesised, no antimonide were present in the MBE system. There is an area of high concentration on the left side of the wurtzite stem, but comparing with the intensity map, this point is of very low intensity, so this should not be given any significance. Above the wurtzite stem starts the InAsSb zincblende alloy. On the right in Figure 10, the wire height is plotted as a function of the mean of the composition rows, to have a measurement on the concentration in each part of the $InAs_{1-x}Sb_x$. These are seen in Table 1 together with the vapour concentrations from the MBE and the results from a recent TEM-EDX analysis of the same growth. The TEM-EDX and the SXDR results are quite similar, although some deviations of the concentration values are present, especially for the highest concentration of antimonide. As we see on the concentration map in Figure 10, these values are quite dependent on where

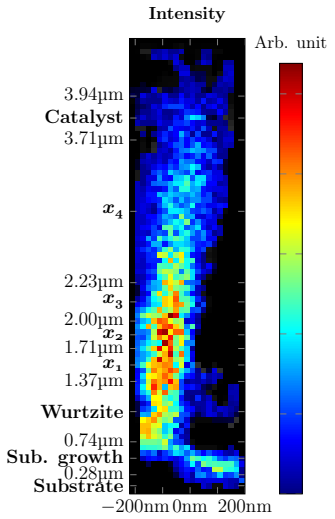


Figure 11: Intensity map. The lower intensities are subject to larger uncertainties in the measurements seen in other plots.

across the wire they are measured. The TEM-EDX might be measured on the side with an apparently higher concentration on the SXDR map, and the SXDR is an average corresponding to the middle of the wire. The TEM-EDX result does not exceed the highest concentration measured with the SXDR.

Region	x	x_v	x_{EDX}
1	0.259	0.15	0.24
2	0.312	0.2	0.29
3	0.376	0.26	0.37
4	0.561	0.55	0.63

Table 1: Table of composition values in each region, x, compared to the vapour concentration from the MBE, x_v and the values measured by TEM-EDX, x_{EDX}.

These measured composition values are all higher than the vapour concentration in the MBE, showing that the incorporation rate of antimonide is higher than that of arsenide. It seems that as the antimonide concentration rises, the incorporated part approaches the vapour concentration. A plot of $x(x_v)$ and $x_{EDX}(x_v)$ is seen in Figure 12. Both sets of experimental values shows the same tendency towards a lower slope at higher concentrations, although not as pronounced on the TEM-EDX. Looking at the interfaces between different sections, these seem gradual. This does not necessarily mean that the interfaces are gradient, since the X-ray beam used for this analysis is big compared to the atomic layers, so the beam is overlapping both sections causing the interface to appear gradual. The same applies to the width of the wire: As soon the beam hits the wire partially, we will have a diffraction signal, so the wire appears a lot wider than it actually is. This is because the X-ray beam is almost as wide as the wire. This situation is sketched on Figure 13, and from this we see that the apparent width is $d_a = d_{NW} + d_{beam}$. This fits well with what is seen on the intensity map: The high intensity region going up the middle of the wire is around the size of the wire measured independently from this SXRD, while there is still some notable intensity beyond this, because the beam partially hits the wire. Due to the same effect, the height will be a bit larger than expected, but since the height to beam size ratio is much larger than the width to beam size ratio, this is not as pronounced. Taking these effects into account, SXDR is not a very suitable method for measuring the dimensions of NWs. Since these are already measured to a high accuracy using TEM-analysis, it is not worth proceeding

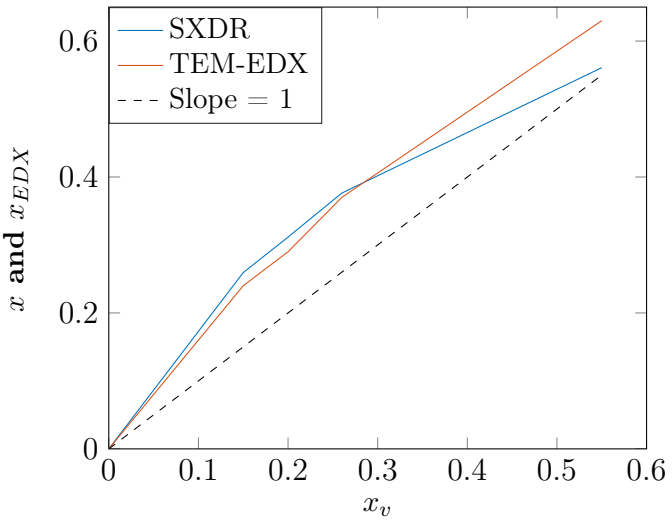


Figure 12: Experimental concentration values plotted. The dashed line is the a linear graph with slope 1, and shows how the curve would look if the incorporated concentration were equal to the vapour concentration.

to calculate these in this characterization, although it would be possible by taking into account the beam size.

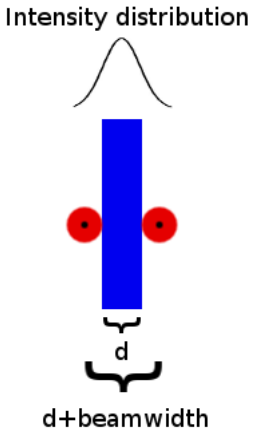


Figure 13: Sketch of the intensity distribution as a function of position across the wire. From this, we can see that the beam adds half a beam width to the apparent width of the nanowire on either side of it. This is what causes the apparently way to wide maps of the wire.

4.1 Strain analysis

One very notable tendency seen in Figure 10, is that the right side of the wire seems to be of lower lattice constant than the left side. This effect seems to be more pronounced with increasing antimonide concentrations. This is a clear indication of an uneven strain distribution caused by the aluminium half shell. To investigate this further, the lattice constants of the wurtzite stem and the four InAsSb sections were calculated separately. This was done by taking the mean of each sections lattice constants. Then, using these mean lattice constants as a reference for the corresponding parts of the wire, the strain was calculated according to equation 1.2. This led to the strain map seen in Figure 14. In the strain map, it is seen how the lower parts of the $InAs_{1-x}Sb_x$ wires are nearly unstrained, but still has a tendency to have tensile strain on the left side of the wire, and compressive strain on the right side. In the top part with the high antimonide concentration, this strain is much more pronounced. Since this is an aluminium half

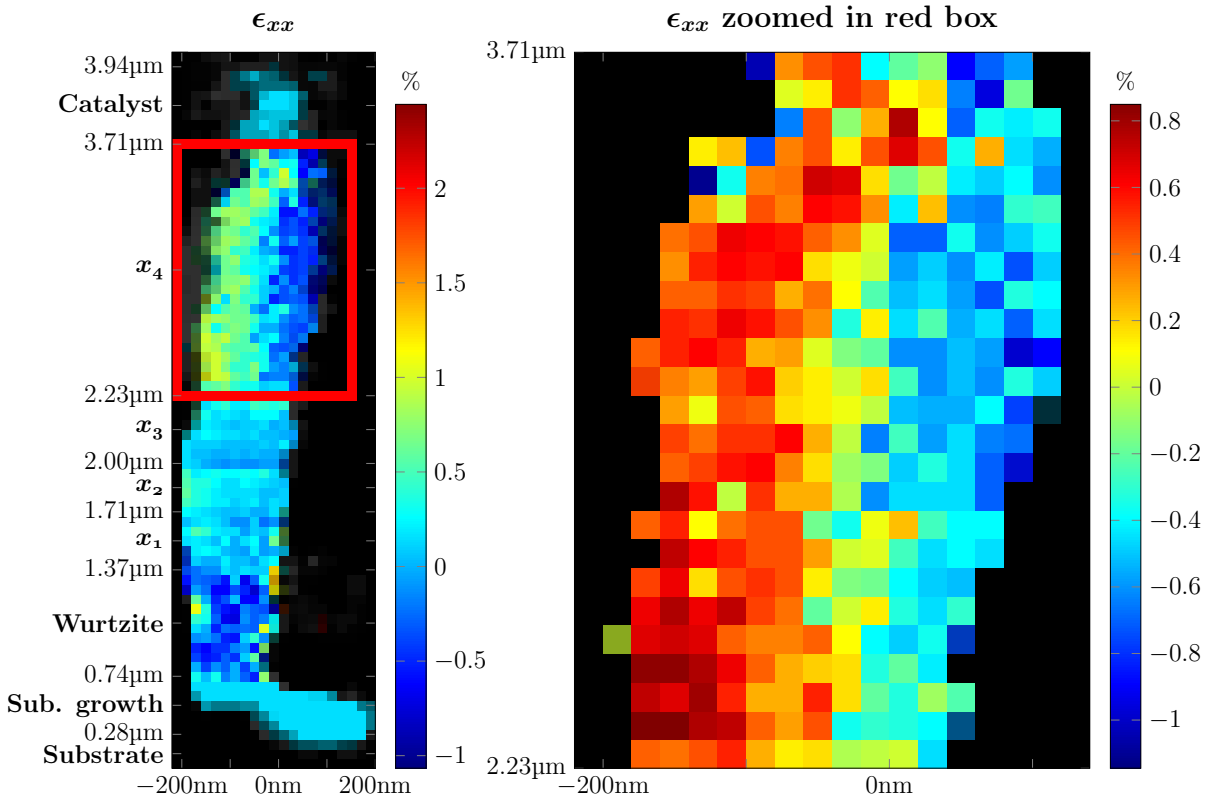


Figure 14: In-plane strainmap of the analysed NW. It is seen that the top part of the wire has an uneven strain distribution. On the right plot there is zoomed in on the top part of the wire. We see that the left side is subjected to a tensile strain and the right side is under a compressive strain. We also se the the NW is relaxed in the middle of the wire.

shell wire, the strain is caused by the interface between core and shell. The even distribution of the strain suggests that it is elastic. If the strain were relaxing plastically, not much strain would have been present in this type of calculation, due to the interface dislocating instead of stretching the crystal[6]. This, however does not seem to be the case. Instead, the aluminium shell causes the strain to distribute radially through the wire causing the unit cells to expand on one side and compress on the other. This is supported by the recent TEM-EDX analysis of the same wire, which shows a very well matched interface between core and shell as seen on Figure 15. The gradual strain distribution across the nanowire is very clearly seen on the zoom in of the upper part of the wire in Figure 14. Unfortunately we were not able to find any Bragg peaks from the aluminium shell, so we are not able to tell if the interface between core and shell causes the tensile or the compressive strain. This is still a very interesting result, since it implies that the strain between core and shell is very dependent on the antimonide concentration.

There seem to be some interfacial strain between the different concentrations of antimonide as well. This is most probably caused by the big beam size, and the different reference values for each part. These means that at these interfaces, the reference value changes instantly from one row to the next, leading to the what looks like interfacial strain, but it is just an artefact of the way the calculations were carried out.

4.2 Summary of results

From the SXDR analysis we found the concentrations of antimonide in different parts of the wire to $x_1 = 0.259$, $x_2 = 0.312$, $x_3 = 0.376$ and $x_4 = 0.561$. By comparing these to the vapour concentrations from the MBE, it suggests that the incorporation rate of antimonide is higher than that of arsenide. Further we found that the aluminium half shell induces strain in the

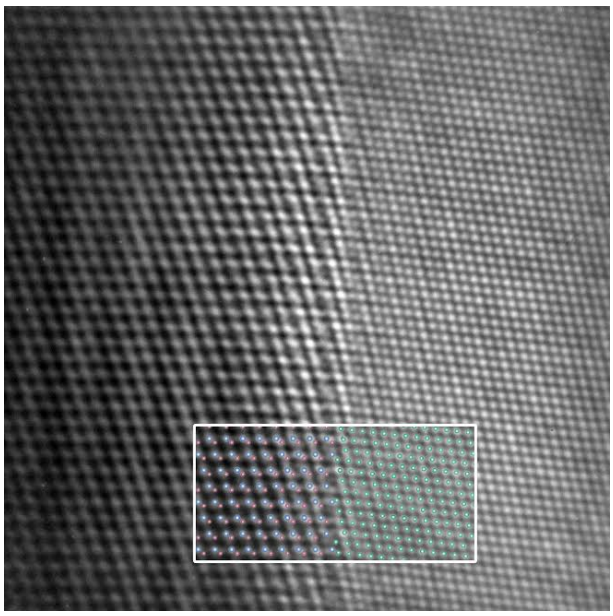


Figure 15: TEM image of the interface between core and shell in the characterized wire. In the box, InAsSb-units and Al atoms are drawn to show how the core and shell are matching. Thanks to Thomas Nordqvist for providing the picture.

nanowire, and that this strain is proportional to the antimonide concentration.

5 Conclusion

In this thesis, the theory of nanowire growth and X-ray characterization has been outlined. Further, some motivation on why this is a topic worth studying has been given, and it is certainly a field in rapid development with interesting discoveries being made often. The theoretical outline is brief but encompasses the central parts of X-ray diffraction and crystallography as well as other related topics. The main focus has been on the experimental work, and a lot of time were spend preparing samples and making measurements at PETRA-III, DESY's synchrotron in Hamburg. These procedures have been described in some detail. The experimental technique scanning X-ray diffraction microscopy has been explained utilizing the theory described in the introduction. A lot of time were spend adapting and developing data analysis scripts (These are included as appendices) for the data gathered from the SXRD experiment, and these were used and tested on a data set from a $InAs_{1-x}Sb_x$ nanowire. This led to the discovery of aluminium half shells straining these types of wires, and that this strain is proportional to the antimonide concentrations. Unfortunately we were not able to tell at which side the aluminium shell was, so we cannot tell if the aluminium induces tensile- or compressive strain. Further, the concentrations of antimonide were found, and these suggested a higher incorporation of antimonide than arsenide for the given MBE growth.

5.1 Outlook

One obvious thing to find out in future analysis is whether the aluminium half shell causes compressive or tensile strain on the high concentrations of antimonide in $InAs_{1-x}Sb_x$, since at the moment this result is not very useful for strain engineering. This could be done by other analyses such as TEM, correlating the bending direction of the wire in the SXDR analysis with the side on which aluminium is present as seen from the TEM. Further, there is still lots of data obtained at DESY still to be analysed, for example from a kinked nanowire. Analysis of this data set could lead to interesting discoveries of strain in the kink, or strain differences between the

horizontal and vertical parts of the wire. Since the data analysis has not been carried out yet, this is just guesswork. Development of new synchrotron beam lines will improve the resolution of future characterizations of the same kind. One major problem found in this analysis was the sample drift, since it made convolution impossible. This is hopefully a problem that will be addressed in the next generation of X-ray nanoprobes such as the MAX IV NanoMAX nanoprobe in Lund, which is still under development. Further, the nano focusing capabilities has improved since the nanoprobe at DESY was developed. NanoMAX aims to a beam size of 10nm[8], so even if the sample drift is still present, the resolution will be improved greatly compared to that at the P06 beamline at DESY and other present synchrotron nanoprobes. These developments will cause future experiments of the same type to hopefully be improved a lot.

References

[1] P06 Nanoprobe at PETRAIII, courtesy of DESY. `http://photon-science.desy.de/facilities/petra_iii/beamlines/p06_hard_x_ray_micro_probe/nanoprobe/index_eng.html`. Accessed: January 28, 2016.

[2] Barkeshli, M., and Sau, J. D. Physical architecture for a universal topological quantum computer based on a network of majorana nanowires. *arXiv preprint arXiv:1509.07135* (2015).

[3] Borg, B. M., and Wernersson, L.-E. Synthesis and properties of antimonide nanowires. *Nanotechnology 24*, 20 (2013), 202001.

[4] Datta, S. Recent advances in high performance CMOS transistors: From planar to non-planar. *Electrochem. Soc. Interface 22* (2013), 41–46.

[5] De Wolf, I., Senez, V., Balboni, R., Armigliato, A., Frabboni, S., Cedola, A., and Lagomarsino, S. Techniques for mechanical strain analysis in sub-micrometer structures: TEM/CBED, micro-Raman spectroscopy, X-ray micro-diffraction and modeling. *Microelectronic engineering 70*, 2 (2003), 425–435.

[6] Dunstan, D. Strain and strain relaxation in semiconductors. *Journal of Materials Science: Materials in Electronics 8*, 6 (1997), 337–375.

[7] Glas, F. Heterostructures and strain relaxation in semiconductor nanowires. *Lattice Engineering: Technology and Applications* (2012), 189.

[8] Johansson, U., Vogt, U., and Mikkelsen, A. NanoMAX: A hard x-ray nanoprobe beamline at MAX IV. In *SPIE Optical Engineering+ Applications* (2013), International Society for Optics and Photonics, pp. 88510L–88510L.

[9] Kittel, C. *Introduction to solid state physics*. Wiley, 2005.

[10] Shiri, D., Kong, Y., Buin, A., and Anantram, M. Strain induced change of bandgap and effective mass in silicon nanowires. *Applied Physics Letters 93*, 7 (2008), 073114.

Appendices

A Matlab Code: Cutting out ROI and saving

Modified script, original provided by Tomas Stankevic.

```
  %clear all
% This program is used to reduce the data to only what is needed and save
% all in one file.

% the scan names can be entered manually or given in a list in a file for
% barch processing

close all
session = '0003_QDev187/'
addpath(genpath('/home/rasmus/Dropbox/Bachelor/Petra Matlab/Petra2015/')); % path
    for matlab programs
vectorpath = '/home/rasmus/Dropbox/Bachelor/Petra_Results/'; % path to save data as
    1 file
scanlist = importdata(fullfile(vectorpath,'scanlist2-10.txt')); % list of scans in a
     text file
datapath = '/home/rasmus/Desktop/Final Data/'; % raw data path

 % Nr of scan from the scan list
%scanNr = 10;
    %scanname = scanlist.textdata{scanNr+1}; display(scanname); % generate scan name
         if from file

    %%%% Manually specify the scan name
    scanname = 'scan_0059';
    snakescan = 1; % if snake scan

    peaks = [365 333]; %; 234 539 ; 200 210] %; 229 498 ; 216 330]; % Specify pixel
        coordinates on detector for the Bragg peaks (one row per peak)
    % peak Nr 2

    logfilepath = fullfile(datapath,session,scanname,strcat(scanname,'.txt')); %
        generate log file path
    imagepath = fullfile(datapath,session,scanname,'ccd','pilatus01'); % generate
        image path
    fluorpath = fullfile(datapath,session,scanname,'maps','mapallXiacounts.edf'); %
        generate fluorescence file path

    logfile = importdata(logfilepath,' ',42); % import logfile
    mkdir(vectorpath,scanname);   savepath = fullfile(vectorpath,scanname); % make
        folder and path for the saving file
    header = logfile.textdata; data = logfile.data; % read log file. important part
        is in "data"
    ioncham = data(:,end); % read ion chamber column from the log file. It measures
        beam intensity before the sample
    ioncham0 = ioncham./5850; % just divide it so it's not such a big number
    [~, fluor] = pmedf_read(fluorpath); % read fluorescence map

    sample_name = 'QDev187'; % sample name ("QdevXXX")

    interferometer_y = logfile.data(:,8);
    interferometer30 = logfile.data(:,10);
    interferometer60 = logfile.data(:,12);

    yses_row = logfile.data(:,3); % y coordinates from log file
```

```
    xses_row = logfile.data(:,4); % x coordinates from log file
45  %
    xses=unique(xses_row); yses = unique(yses_row); % unique values
47  %yses(end)=[];

49  cols=0:size(xses,1)-1; % generate column and row numbers
    rows=0:size(yses,1)-1; % generate column and row numbers
51  ioncham =ioncham0(1:(length(xses)*(length(yses)))); % take only the ionchamber
        values for which we have diffraction measurement


53
    ioncham = (reshape(ioncham,size(xses,1),size(yses,1)))'; % reshape it to the
        scan matrix size
55  % if snakescan, flip every second row
    if snakescan
57      even=  ~mod(1:size(yses,1),2);
        ioncham(even,:)=fliplr(ioncham(even,:));
59      %fluor=fluor'./(ioncham).^2;
    end
61
    clear ROI
63




65



67
    names = {'$$(101)$$'} %,'$$(1 0 1)$$', '$$3rd guy$$'}; % Names of the
        corresponding peaks
69



71
    ROI_size = 80; % size in pixels of region of interest(ROI) (+/-). 40 means ROI
        will be 81x81
73  % generate ROI boundaries
    for i=1:length(peaks(:,1))
75      ROI(i,1:2)=peaks(i,1:2)-ROI_size;
        ROI(i,3:4)=peaks(i,1:2)+ROI_size;
77  end;

79  %start = cols(1); % start column
    %ending = cols(2); % end column
81  %scan_z = yses; %mm, mm, intervals
    %scan_y = xses; %mm, mm, intervals
83
    %step_z = (scan_z(2)-scan_z(1))/scan_z(3);
85  %step_y = (scan_y(2)-scan_y(1))/scan_y(3);

87  %xses = (cols)*step_y*1000;
    %new_yses = (rows)*step_z*1000;
89
    %}
91
    new_yses = yses; % just a new variable for Y
93
    Energy = 18; % keV
95  Lambda = 12.39842/Energy; % wavelength

97  X0 = 244; % coordinates of the direct beam on detector (pixels)
    Y0 = 310; % coordinates of the direct beam on detector (pixels)
99
    bcgc = [51 51]; % coordinates of an empty area on the detector for background
        subtraction
101 Name = 'Diffraction'; xlab=''; ylab=''; tit=['']; % just plot parameters (name
```

```
    etc)

% plot one image to check. Make sure that the numbers exist!
rrr = 70; ccc = 9; % row and column of a random existing image
no1 = sprintf('%04d', rrr); % generate file name. row number
no2 = sprintf('%05d', ccc); % generate file name. col number
no = strcat('ccd_',no1,'_',no2,'.edf'); % combine together
actualImage = fullfile(imagepath,no); % full file name
[header, image] = pmedf_read(actualImage); % read file
%image = double(imread(actualImage));
%figure(11);
%clf;

NiceFig([8.46, 8.46], Name, 8, savepath, [], [], log10(image'), xlab, ylab, tit,
     false, false) % plot diffraction map
colormap('hot'); daspect([1 1 1]); drawnow; hold all;

% plot rectangles showing ROI
for t=1:length(names)
    rectangle('Position',[ROI(t,1),ROI(t,2),ROI(t,3)-ROI(t,1),ROI(t,4)-ROI(t,2)
        ],'EdgeColor','w');
    text(peaks(t,1)+50,peaks(t,2)+0,names(t),'Color','w','interpreter','latex','
        FontName','Arial')
end;

% rectangle('Position',[X0-20,Y0-20,40,40],'EdgeColor','w');
% text(X0-250,Y0-150,'Direct beam','Color','w','FontName','Arial');
set(gca,'XTick',[1,487])
set(gca,'XTickLabel',{'1','487'})
set(gca,'YTick',[1,619])
set(gca,'YTickLabel',{'1','619'})

% plot fluorescence map
figure(6223);imagesc(fluor')
plotting = true;

%% Run through all the images, cutting out ROI and saving

% initialize varibles
new_vector = zeros(length(rows),length(cols),length(names),ROI_size*2+1,ROI_size
    *2+1); % vector for the images
distx = zeros(length(names),ROI_size*2+1,ROI_size*2+1); % distances of each
    pixel from the center
disty = zeros(length(names),ROI_size*2+1,ROI_size*2+1); % distances of each
    pixel from the center
im_ROI = zeros(ROI_size*2+1,ROI_size*2+1,length(names)); % ROI of one image
figure(1234);clf; % prepare figure
im_BP = zeros(size(fluor')); % empty image to show fluorescence map
    progressively
imh = imagesc(im_BP); % plot it
hotimage=0;
bcgsize = 50;

for i=rows % cycle through rows
    rowflag = 0; % some flag checking missing images
    for j=cols % cycle through columns
        jj=j-cols(1)+1; % counters
        ii=i-rows(1)+1; % counters
        no1 = sprintf('%04d', i); % generate file name
        no2 = sprintf('%05d', j);
        no = strcat('ccd_',no1,'_',no2,'.edf');
        display(no); % show number
        actualImage = fullfile(imagepath,no);
```

```
157             % image = double(imread(actualImage));
                [header, imagen] = pmedf_read(actualImage); % read image
159
                if header==-1 % if an image is missing, raise a flag
161                 display('Image not found')
                    imageo=imageo; % take a previous image instead
163                 flag=1; rowflag = 1; % raise a flag
                else
165                 imageo = imagen; % if not missing, take a new image
                    flag=0;
167             end
                image=imageo'; % transpose
169             %image = image;%./ioncham(ii,jj); % divide by ion chamber measurement to
                     normalize for beam fluctuations
                image(image>50000) = NaN;
171             figure(1);  clf; % clear figure
                imagesc(log10(image+1)); % plot image
173
                bcg = image(bcgc(1)-bcgsize:bcgc(1)+bcgsize,bcgc(2)-bcgsize:bcgc(2)+
                    bcgsize); % take background
175             bcgm = mean(mean(bcg)); % mean it
                bcgs = mean(std(bcg)); % std of background
177             if bcgs>1000
                    disp(bcgs);
179                 hotimage=1;
                else
181                 hotimage=0;
                end
183             for k=1:length(names) % cycle through bragg peaks
                    [X,Y] = meshgrid(ROI(k,1):ROI(k,3),ROI(k,2):ROI(k,4)); % generate
                        pixel coordinates
185                 X=X-X0; % distance from center
                    Y=Y-Y0; % distance from center
187                 im_ROIo = image(ROI(k,2):ROI(k,4),ROI(k,1):ROI(k,3));%-bcgm; % take
                        the ROI
                    %im_ROIo(im_ROIo<3*bcgs) = 0;
189                 im_ROI(:,:,k)=im_ROIo; % put it in a vector
                    new_vector(ii,jj,k,:,:)=im_ROIo; % and then in a bigger vector
191                 distx(k,:,:)=X; % put distances in their own vectors
                    disty(k,:,:)=Y; % put distances in their own vectors
193                 % something to deal with missing images. let's hope it doesn't
                        happen


195
                    if mod(ii,2)
197                     im_BP(ii,jj) = sum(im_ROIo(:));
                    else
199                     im_BP(ii,end-jj+1) = fliplr(sum(im_ROIo(:)));
                    end
201
                    if hotimage
203                     new_vector(ii,jj,k,:,:)=0;
                    end
205


207             end % end of Bragg peak loop
                % Plot image (update plot with new data)
209             set(imh,'CData',im_BP);
            end % end of column loop
211         % something to deal with missing images. let's hope it doesn't happen
            if rowflag
213             if mod(ii,2)
                    im_BP(ii,:)=circshift(im_BP(ii,:),1,2);
```

```
                new_vector(ii,:,:,:,:) = circshift(new_vector(ii,:,:,:,:),1,2);
            else
                im_BP(ii,:)=circshift(im_BP(ii,:),-1,2);
                new_vector(ii,:,:,:,:) = circshift(new_vector(ii,:,:,:,:),1,2);
            end
            set(imh,'CData',im_BP);
        end

        % the end
    end % end of row loop



    %% Don't forget to SAVE

    vector = new_vector; % some shuffle, so that you don't immediately overwrite old
         vector with new
    yses = new_yses; % some shuffle, so that you don't immediately overwrite old
        vector with new
    filepath = fullfile(savepath); % file path
    filepath = fullfile(filepath,strcat(sample_name,'.mat')); % file path
    save(filepath,'vector','distx','disty','xses','yses','names','fluor','
        interferometer_y','interferometer30','interferometer60'); % save all needed
        variables in 1 file
    display('Vector Saved')

    figure(scanNr+1)
    imagesc(squeeze(nanmean(nanmean(log(vector(:,:,1,:,:)+1)))));


%%
%clear all;
```

B Matlab Code: Correlating Intensities with real space positions

Modified script, original provided by Tomas Stankevic.

```
addpath(genpath('/home/rasmus/Dropbox/Petra Matlab/Petra2015/')); % path for matlab programs
% scanname = 'scan_0313'; % scan name
%samplename = 'QDev90'; % sample name
scanname = 'scan_0082';
%scanname = 'scan_0060';
samplename = 'QDev187';
snakescan = 1;
filepath = '/home/rasmus/Dropbox/Bachelor/Petra_Results/';  % path to vector files
filepath = fullfile(filepath,scanname,strcat(samplename,'.mat')); % gen file path
load(filepath);
%vector1 = vector; % load vector %Since we don't unload the vector file,
%there's no need to take it out of the file.

fluor = fluor'; % transpose fluorescence

%%
k=1; % Choose Bragg peak number

yrange = 1:size(vector,1);
%yrange = 1:55;
%yrange = 1:44
%yrange = 59:size(vector,1);

Imean = squeeze(nanmean(nanmean(vector(yrange,:,k,:,:),1),2)); % take it out of the vector and average over all scan points for plotting (1 and 2 dimensions)

figure(1); clf; subplot(1,2,1); % prepare figure
imBP = imagesc(log10(Imean));  % plot average Bragg peak

hp = impoly(); % make a draggable polygon
title('Bragg peak, Log(I), scan 0090');

Inw = squeeze(nanmean(nanmean(vector(yrange,:,k,:,:),4),5)); % mean all bragg peak points to image nanowire (dimnesions 4 and 5)

if snakescan % if snakescan flip every second row
    for i = 2:2:length(Inw(:,1))
        Inw(i,:) = fliplr(Inw(i,:));
    end
end
subplot(1,2,2); imNW = imagesc(log10(Inw)); title('Intensity');daspect([3,2,1]); colorbar; % plot integrated bragg peak  - nanowire image

I = vector(:,:,k,:,:); % Now take the whole data for the given Bragg peak
siz=size(vector);
BPsize=siz(4:5);
while 1
    I = vector(:,:,k,:,:); % Now take the whole data for the given Bragg peak
    BW = createMask(hp); % make a mask out of the polygon

    BWrep = repmat(reshape(BW,1,1,1,BPsize(1),BPsize(2)),size(vector,1),size(vector,2),1,1,1); % repeat it many times so that it has the same dimensions as the vector
    I = I.*BWrep; % multiply mask (ROI) and intensity
    I(isnan(I))=0; % not a number intensity make zero

    Inw =sum(sum(I,4),5); % sum the intensity within ROI
```

```
    % if snakescan flip every second row
    if snakescan
        for i = 2:2:length(Inw(:,1))
            Inw(i,:) = fliplr(Inw(i,:));
        end
    end

    set(imNW,'Cdata',Inw(1:end,:)); drawnow; % update plot
end

%% Find Center of mass and plot weighted by intensity
I(isnan(I))=0; % make shure there is no NaN
Mass = sum(sum(I,4),5); % Total mass

Xses = 1:(size(vector,5)); % pixel coordinates
Yses = 1:(size(vector,5)); % pixel coordinates
[X,Y] = meshgrid(Xses,Yses); % pixel coord grid
X = repmat(reshape(X,1,1,1,BPsize(1),BPsize(2)),size(vector,1),size(vector,2),1,1,1)
    ; % repeat
Y = repmat(reshape(Y,1,1,1,BPsize(1),BPsize(2)),size(vector,1),size(vector,2),1,1,1)
    ; % repeat

XM = squeeze(X.*I);     COMx = sum(sum(XM,3),4)./(Mass); % COM X
YM = squeeze(Y.*I);     COMy = sum(sum(YM,3),4)./(Mass); % COM Y

if snakescan % flip if snakescan
for i = 2:2:length(COMx(:,1))
    COMx(i,:) = fliplr(COMx(i,:));
    COMy(i,:) = fliplr(COMy(i,:));
    Mass(i,:) = fliplr(Mass(i,:));
end
end

figure(16);clf;set(gcf,'renderer','opengl'); % prep figure

subplot(2,2,1)
threshold = 1; % threshold for dimming low intensity features (>=1)
P3_plot_weighted(fluor,fluor,'Fluorescence',threshold) % plot fluorescence, no need
    for dimming too much

subplot(2,2,2)
P3_plot_weighted(Mass,Mass,'Diffraction',threshold) % plot peak intensity map

threshold = 1; % threshold for dimming low intensity features (>=1)
subplot(2,2,3)
P3_plot_weighted(Mass,COMx,'COM_x weighted by Intensity',threshold) % plot COMx

subplot(2,2,4); cla
set(gcf,'renderer','OpenGL');
P3_plot_weighted(Mass,COMy,'COM_y weighted by Intensity',threshold) % plot COMy


%% Save COMs and diffraction maps
% fluor=fluor(1:end-5,:);
% Mass=Mass(1:end-5,:);
% COMx=COMx(1:end-5,:);
% COMy=COMy(1:end-5,:);


vectorpath = '/home/rasmus/Desktop/Petra_Results';
savepath = fullfile(vectorpath,scanname);

filepath = fullfile(savepath); % file path
```

```
113 filepath = fullfile(filepath,strcat(samplename,'_COMs_diffraction.mat')); % file
        path
    save(filepath,'COMy','COMx','Mass');
```

C Matlab Code: Calculating pixel coordinates

Modified script, original provided by Tomas Stankevic.

```
function [H, K, L] = P3_calcHKL(distx ,disty, detx, dety, detz, detr, pixel_size,
    Lambda, omega0)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function for calculating normalized reciprocal space coordinates (H,K,L)
% for each detector pixel. Reciprocal space coordinates are calculated from
% the experimental geometry (detector positions) and normalized with respect
% to a given ideal crystal lattice.
% Function assumes horizontal scattering geometry and uses transformation
% matrix described in Schleputz, Mariager et al.
% Output: H, K, L - vectors containing reciprocal space coordinates
% Input arguments>
% distx, disty - pixel position on the detector with respect to the center
%                pixel. Center pixel is determined by the direct beam.
%                distx, disty are measured in pixels, can be fractional.
% det_distance - sample-detector distance in mm
% Lambda       - wavelength in
% omega0       - sample azimuth

% Important!: detx, dety, detz, detr - actual detector table offsets from
% the direct beam position in mm, deg, not raw detector motor values.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

addpath(genpath('/home/rasmus/Dropbox/Bachelor/Petra Matlab/P3_last'));
% Lattice constants for orthonormalization. Comment-out which not needed
% GaN
%aLat = 3.189; bLat = 3.189; cLat = 5.1825;

% InAs

aCub = 6.0583;  % Cubic
aLat = aCub*sqrt(2)/2; bLat = aCub*sqrt(2)/2; cLat = aCub*sqrt(3); % Hexagonal (
    surface)

N=[0 0 1];          % Surface normal
a1 = [aLat   0   0];                          % Lattice vectors. a1
a2 = [aLat*cosd(120)  bLat*sind(120)   0];    % Lattice vectors. a2
a3 = [0   0   cLat];                          % Lattice vectors. a3
% Calculates length and angles between real lattice vectors (a, aa) and
% reciprocal lattice vectors (b, ba)
[a, aa, b, ba] = aps_lengthAndAngles(a1,a2,a3);
% Calculates the normalization matrix B. B consists of columns comprising
% the reciprocal lattice vectors.
[~,~,~,B] = aps_cartesian(N(1),N(2),N(3),a,aa,b,ba);
% Calculate K vector
Kvec=2*pi/Lambda;

% Sample orientation angles. Used in orientation matrix and
% need to be adjusted manually for each sample
% Adjust these angles after looking at the BP position in reciprocal space

alpha=0;%2; %Rotation around x
beta=1; %Rotation around y
gamma=0;%.5; %Rotation around z


%%
    % Alternative solution from AutoCAD drawing
    det_distance = detx; % 520.3000; % sample detector distance in direction of
```

```
        direct beam
    theta = detr; %(17 deg)
    table_radius = 408.0; %mm
    shiftx = dety; %280; %mm detector table shift in horizontal plane from direct
        beam position perp to the direct beam
    shifty = detz; % 0 ; detector offset from direct beam position in detz, or y in
        detector plane, (vertical axis)
    One = ones(size(distx(:))); % Matrix of "1" of size of the detector image

    % Pixel coordinates with respect to the center of detector table at
    % reference position of the direct beam (not rotated, detector plane perp to the
         direct beam)
    % z is parallel to the beam, xy in detector plane
    xyz_table = [pixel_size*distx(:), pixel_size*disty(:), -One.*table_radius]';

    % Now rotate the table around the vertical axis (Y).
    % Rotation matrix around Y:
    Ry = [cosd(theta) 0  sind(theta);
                0     1      0;
         -sind(theta) 0  cosd(theta)];
    xyz_table = Ry*xyz_table;

    % Move the table into the laboratory frame
    xyz_lab = xyz_table + [One.*shiftx, One.*shifty, One.*(det_distance +
        table_radius)]';

    xyz_lab = reshape(xyz_lab,[3,size(distx)]); % Reshape to the size of 3 columns

    Xp = squeeze(xyz_lab(1,:,:,:)); % X-coordinate of each pixel in laboratory frame
    Yp = squeeze(xyz_lab(2,:,:,:)); % Y-coordinate of each pixel in laboratory frame
    Zp = squeeze(xyz_lab(3,:,:,:)); % Z-coordinate of each pixel in laboratory frame
    Rp = sqrt(squeeze(sum(xyz_lab.^2,1))); % Sample-pixel distance for each pixel

    Gam = -atan2(Xp,Zp); % Angle Gamma (horizontal plane) in radians
    Del = -asin(Yp./Rp); % Angle Delta (vertical plane) in radians

% Sample orientation matrix U. Bounds the sample crystal with the laboratory frame
% Angles alpha beta gamma were manually adjusted so that known peaks
% are exactly in their places

% X is horizontal, perp to the beam, Y is vertical

Rx = [1         0            0; % Sample rotation around X (alpha)
    0 cosd(alpha) -sind(alpha);
    0 sind(alpha)  cosd(alpha)];

Ry = [cosd(beta)  0  sind(beta); % Sample rotation around Y (beta)
    0        1       0;
    -sind(beta)  0  cosd(beta)];

Rz = [cosd(gamma)  -sind(gamma) 0; % sample rotation around Z (gamma)
    sind(gamma)  cosd(gamma)  0;
    0         0         1];

U = Rx*Ry*Rz;

% multiply with normalization matrix B to get HKL
UB = U*B;

ov = omega0/180*pi; % convert to radians
alp = 0; % incidence angle between direct beam and substrate 0

M1 = Kvec.*(-cos(ov).*sin(Gam).*cos(Del)+...
```

```
    sin(ov).*(cos(alp).*(cos(Gam).*cos(Del)-1)+sin(alp).*sin(Del)));
M2 = Kvec.*( sin(ov).*sin(Gam).*cos(Del)+...
    cos(ov).*(cos(alp).*(cos(Gam).*cos(Del)-1)+sin(alp).*sin(Del)));
M3 = Kvec.*( -sin(alp).*(cos(Gam).*cos(Del)-1)+cos(alp).*sin(Del));

% If the matrix cannot be inverted you have a problem.
UBi = inv(UB);

% Now calculate the hkl in reciprocal lattice coordinates.
H = UBi(1,1)*M1 + UBi(1,2)*M2 + UBi(1,3)*M3;
K = UBi(2,1)*M1 + UBi(2,2)*M2 + UBi(2,3)*M3;
L = UBi(3,1)*M1 + UBi(3,2)*M2 + UBi(3,3)*M3;
```

D Matlab Code: Calculating COMs, composition and strain

Modified script, original provided by Tomas Stankevic.

```
%function P3_plot5D()

clear all
addpath(genpath('/home/rasmus/Dropbox/Bachelor/Petra Matlab/P3_last'));
addpath(genpath('/home/rasmus/Dropbox/Bachelor/Petra Matlab/P3_last/src'));

vectorpath = '/home/rasmus/Dropbox/Bachelor/Petra_Results';       % path to vectors
sample_name = 'QDev187';
scanlist = importdata(fullfile(vectorpath,'scanlist2-10.txt'));
s2c = [1 1/2 0; 0 sqrt(3)/2 0; 0 0 1];

Recipsize=[161 1 161]; % size of reciprocal space
%Recipsize=[137 1 137];
Realsize = [71 21 size(scanlist.data,1)]; %2-10
%Realsize = [70 10 size(scanlist.data,1)]; %101

Megavector_in = {1};
Megavector_in{1} = zeros(Realsize(3),Realsize(1),Realsize(2),Recipsize(1),Recipsize
    (3));
%Megavector_in{2} = zeros(Realsize(3),Realsize(1),Realsize(2),81,81);


%% Load megavector_in


for scan_Nr = 4:14%14
    display(num2str(scan_Nr));
    load(fullfile(vectorpath,[scanlist.textdata{scan_Nr+1}],strcat(sample_name,'.mat
        ')),'vector','names'); % load each vector
    for peakNr = 1:length(names)
        I = squeeze(vector(1:Realsize(1),1:Realsize(2),peakNr,:,:));
        bcg=nanmean(nanmean(nanmean(nanmean(I(:,:,1:10,1:10)))));
        I=I-bcg;
        I(I<0)=0;
        Megavector_in{peakNr}(scan_Nr,1:size(I,1),1:size(I,2),1:size(I,3),1:size(I
            ,3))=I;
    end
end
% load flatfield
% load(fullfile(vectorpath,'ccd_ff.mat'));


```

```
%% Load coordinates HKL and interpolate mean BP
%xses = {1,1};yses = {1,1};zses = {1,1};
XYZ_all = {1,1};


for peakNr = 1:length(names)
    clear H K L HKL
    % Set interpolation limits
    %siz=121; rangg=0.075;
    %if peakNr==1
    %    xses{peakNr} = linspace(0.98-rangg,0.98+rangg,Recipsize(1));
    %    yses{peakNr} = linspace(0.005-rangg/4,0.005+rangg/4,Recipsize(2));
    %    zses{peakNr} = linspace(-rangg,rangg,Recipsize(3));
    %else
    %    xses{peakNr} = linspace(0.97-rangg,0.97+rangg,Recipsize(1));
    %    yses{peakNr} = linspace(0.004-rangg/4,0.004+rangg/4,Recipsize(2));
    %    zses{peakNr} = linspace(0.97-rangg,0.97+rangg,Recipsize(3));
    %end
    %[xm, ym, zm] = meshgrid(xses{peakNr},yses{peakNr},zses{peakNr});

    XYZ_all{peakNr} = zeros(3,Realsize(3),Recipsize(1,1,1)^2);
    for scan_Nr = 1:Realsize(3)
        scanpath = fullfile(vectorpath,scanlist.textdata{scan_Nr+1});
        filepath = fullfile(scanpath,strcat('HKLvector.mat'));
        load(filepath,'H','K','L');
        if ndims(H) == 2;
            HKL = zeros(3,length(H)^2);
            HKL(1,:) = reshape(H(:,:),1,[]);
            HKL(2,:) = reshape(K(:,:),1,[]);
            HKL(3,:) = reshape(L(:,:),1,[]);
        else
            HKL = zeros(3,length(H)^2);
            HKL(1,:) = reshape(squeeze(H(peakNr,:,:)),1,[]);
            HKL(2,:) = reshape(squeeze(K(peakNr,:,:)),1,[]);
            HKL(3,:) = reshape(squeeze(L(peakNr,:,:)),1,[]);
        end
        XYZ_all{peakNr}(:,scan_Nr,1:length(HKL)) = s2c*HKL;   % convert to cartesian
        %Hp(scan_Nr,:,:)=H(peakNr,:,:);
        %Kp(scan_Nr,:,:)=K(peakNr,:,:);
        %Lp(scan_Nr,:,:)=L(peakNr,:,:);
    end
    % XYZp = reshape(XYZ_all{peakNr},3,Realsize(3),161,161);

end

clear HKL H K L


%% COM CALCULATIONS
    ##########################################################################################


% Initialize some variables
snakescan = 1;
%COMXref = {1,1}; COMYref = {1,1};COMZref = {1,1};
%COMX = {1,1}; COMY = {1,1};COMZ = {1,1};COMR = {1,1}; %Megavector_inp = {1,1};
COMX = zeros(Realsize(1),Realsize(2),Realsize(3));
COMY = COMX; COMZ = COMX; COMR = COMX; Ome = COMX; Phi = COMX; Mass = COMX;
Imean = zeros(Recipsize(1),Recipsize(3),Realsize(3));
I1D = zeros(Recipsize(1)*Recipsize(3),Realsize(3)); cutindices = I1D; cutindices2 =
    I1D; cutindices3 = I1D;
indx = I1D'; indy = I1D'; indz = I1D'; indcutx= I1D'; indcuty = I1D'; indcutz = I1D
```

```
    ';


%  PEAK NR 1

peakNr = 1;

% LIST OF SCANS TO INCLUDE
%scans_incl = [1,2,3,4,5,6,8,9,10];
scans_incl = [4,5,8,11,13,14];
%scans_incl = [4,5,8,11,13,14];
%scans_incl = [10];
%scans_incl = [5,6,7,8,9,10,11,12,13,14];
%scans_incl = [3,5,7,9,11,13];
%scans_incl = [1,2,3,4,5,6,7,8,9,10,11,12,13];
%scans_incl = [2,3,4,5,6,8,9,10];
%scans_incl = [2,3,4,5,6,7,8,9,10];
cutoff = 0;

for scans = 1:length(scans_incl)
    scans = scans_incl(scans)

    Imean = squeeze(nanmean(nanmean(Megavector_in{peakNr}(scans,:,:,:,:),2),3));
    %Megavector_in{peakNr}(:,:,scans_incl,:) = reshape(permute(Megavector_in{peakNr
        }(scans_incl,:,:,:,:),[2,3,1,4,5]),Realsize(1),Realsize(2),1,[]);
    Megavector = reshape(permute(Megavector_in{peakNr}(scans,:,:,:,:),[2,3,1,4,5]),
        Realsize(1),Realsize(2),[]);
    XYZ = XYZ_all{peakNr}(:,scans,:);
    XYZr(scans,:,:) = reshape(XYZ,3,size(XYZ,2)*size(XYZ,3)); %clear HKL;

    %clear XYZ_all


    %%%%%%%%%%%  ROI for peak 1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Qxref = [1.45 1.5]; Qyref = [-0.88 -0.82]; Qzref = [-0.15 0.05];
    %Full peak ->
    %if scans == 14;
    %    Qxref = [1.42 1.435]; Qyref = [-0.84 -0.81]; Qzref = [-0.07 -0.005];
    %else
        Qxref = [1.42 1.6]; Qyref = [-0.95 -0.81]; Qzref = [-0.07 0.1];
    %end
    %Substrate peak only ->
    %Qxref = [1.475 1.6]; Qyref = [-0.95 -0.845]; Qzref = [-0.025 0.02];
    %High konc peak, scan 14

    %Qxref = [1.35 1.45]; Qyref = [-0.82 -0.78]; Qzref = [-0.15 0.1];
    %Qxref = [1.45 1.5]; Qyref = [-0.95 -0.83]; Qzref = [-0.15 0.1];
    %Qxref = [1.485 1.6]; Qyref = [-0.95 -0.84]; Qzref = [-0.01 0.01];
    %Second substrate peak ->
    Qxcut = [1.4 1.47]; Qycut = [-0.84 -0.8]; Qzcut = [-0.01 0.2];
    %Wurtzite stem ->
    %Qxcut2 = [1.45 1.55]; Qycut2 = [-0.9 -0.85]; Qzcut2 = [-0.08 -0.022];
    %Noise ->
    Qxcut3 = [1.44 1.45]; Qycut3 = [-0.84 -0.815]; Qzcut3 = [-0.1 0];

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    indcutx(scans,:) = XYZr(scans,1,:)>Qxcut(1)& XYZr(scans,1,:)<Qxcut(2);
    indcuty(scans,:) = XYZr(scans,2,:)>Qycut(1)& XYZr(scans,2,:)<Qycut(2);
    indcutz(scans,:) = XYZr(scans,3,:)>Qzcut(1)& XYZr(scans,3,:)<Qzcut(2);

%     indcutx2(scans,:) = XYZr(scans,1,:)>Qxcut2(1)& XYZr(scans,1,:)<Qxcut2(2);
%     indcuty2(scans,:) = XYZr(scans,2,:)>Qycut2(1)& XYZr(scans,2,:)<Qycut2(2);
```

```
%     indcutz2(scans,:) = XYZr(scans,3,:)>Qzcut2(1)& XYZr(scans,3,:)<Qzcut2(2);
162
    indcutx3(scans,:) = XYZr(scans,1,:)>Qxcut3(1)& XYZr(scans,1,:)<Qxcut3(2);
164 indcuty3(scans,:) = XYZr(scans,2,:)>Qycut3(1)& XYZr(scans,2,:)<Qycut3(2);
    indcutz3(scans,:) = XYZr(scans,3,:)>Qzcut3(1)& XYZr(scans,3,:)<Qzcut3(2);
166

168 I1D(:,scans)=reshape(Imean(:,:),size(XYZ,2)*size(XYZ,3),[]);

170 if cutoff == 1
        cutindices(:,scans) = squeeze(indcutx(scans,:)) & squeeze(indcuty(scans,:))
            & squeeze(indcutz(scans,:));
172     %cutindices2(:,scans) = squeeze(indcutx2(scans,:)) & squeeze(indcuty2(scans
            ,:)) & squeeze(indcutz2(scans,:));
        cutindices3(:,scans) = squeeze(indcutx3(scans,:)) & squeeze(indcuty3(scans
            ,:)) & squeeze(indcutz3(scans,:));
174 end

176

178 indices=(I1D)>1.5*bcg;
    I1D(I1D>20)=20;
180 %Indices for ROI
    indx(scans,:) = XYZr(scans,1,:)>Qxref(1)&XYZr(scans,1,:)<Qxref(2);
182 indy(scans,:) = XYZr(scans,2,:)>Qyref(1)&XYZr(scans,2,:)<Qyref(2);
    indz(scans,:) = XYZr(scans,3,:)>Qzref(1)&XYZr(scans,3,:)<Qzref(2);
184 indices = indices' & ~cutindices' & ~cutindices3' & squeeze(indx) & squeeze(indy
        ) & squeeze(indz);

186 %Normalize intensity
%     for i = 1:Realsize(1)
188 %         Megavector(i,:,:) = Megavector(i,:,:)./nansum(nansum(Megavector(i,:,:)));
%     end
190

    %Indices for cutout
192 %     figure(555);clf;  % plot 3D
%     scatter3(XYZr{peakNr}(1,indices),XYZr{peakNr}(2,indices),XYZr{peakNr}(3,
    indices),60,(I1D(indices)),'fill','s'); hold on; axis equal;
194 %     xlabel('x');ylabel('y');zlabel('z');
%     drawnow;
196 % COM calculations for peak 1
    %[Mass{peakNr}, COMX{peakNr}, COMY{peakNr}, COMZ{peakNr}, COMR{peakNr}, Ome, Phi
        ] = P3_findCOM3D(indices,Megavector_in{peakNr},XYZ{peakNr},Realsize);
198 [Mass(:,:,scans), COMX(:,:,scans), COMY(:,:,scans), COMZ(:,:,scans), COMR(:,:,
        scans), Ome(:,:,scans), Phi(:,:,scans)] = P3_findCOM3D(indices(scans,:),
        Megavector,XYZ,Realsize);

200 %EPSZZ_1 = (-0.09./(COMZ{peakNr})-1).*100;
    %EPSYY_1 = (-0.8475./(COMY{peakNr})-1).*100;
202

204 if snakescan ==1
        %    for peaknr = 1:3
        for i = 2:2:length(Mass)
206         Mass(i,:,scans) = fliplr(Mass(i,:,scans));
            COMX(i,:,scans) = fliplr(COMX(i,:,scans));
208         COMY(i,:,scans) = fliplr(COMY(i,:,scans));
            COMZ(i,:,scans) = fliplr(COMZ(i,:,scans));
210         COMR(i,:,scans) = fliplr(COMR(i,:,scans));
            Ome(i,:,scans) = fliplr(Ome(i,:,scans));
212         Phi(i,:) = fliplr(Phi(i,:));
            %EPSXX_1(i,:,scans_incl) = fliplr(EPSXX_1(i,:,scans_incl));
214         %EPSZZ_1(i,:) = fliplr(EPSZZ_1(i,:));
            %EPSYY_1(i,:) = fliplr(EPSYY_1(i,:));
```

```
        end
        %    end
    end
    %EPSXX_1(:,:,scans) = (1.481./(COMX(:,:,scans))-1).*100;
    %EPSXX_1(:,:,scans) = (1.486./(COMX(:,:,scans))-1).*100;
    %EPSXX_1(:,:,scans) = (1.7218473./(COMR(:,:,scans))-1).*100;
    COMvert(:,:,scans) = sqrt(COMX(:,:,scans).^2+COMY(:,:,scans).^2);



    clear Megavector XYZ
end



% figure(3587); imagesc(Mass{peakNr}(:,:))
%
% figure(111); clf; set(gcf,'renderer','OpenGL'); peakNr = 1; refrow = 5;
% %subplot(2,8,1);  P3_plot_weighted (Mass{peakNr},Mass{peakNr},'Mass',refrow);set(gca,'FontSize',13)
% subplot(1,8,1);  P3_plot_weighted (Mass{peakNr},COMX{peakNr},'COM(Q_x)',refrow); set(gca,'FontSize',13)
% subplot(1,8,2);  P3_plot_weighted (Mass{peakNr},COMY{peakNr},'COM(Q_y)',refrow); set(gca,'FontSize',13)
% subplot(1,8,3);  P3_plot_weighted (Mass{peakNr},COMZ{peakNr},'COM(Q_z)',refrow); set(gca,'FontSize',13)
% subplot(1,8,4);  P3_plot_weighted (Mass{peakNr},COMR{peakNr},'COM(Q_R)',refrow); set(gca,'FontSize',13)
% %subplot(2,8,6);  P3_plot_weighted (Mass{peakNr},EPSYY_1,   '\epsilon _{yy}', refrow);set(gca,'FontSize',13)
% %subplot(2,8,7);  P3_plot_weighted (Mass{peakNr},EPSZZ_1,   '\epsilon _{zz}', refrow);set(gca,'FontSize',13)
% subplot(1,8,5);  P3_plot_weighted (Mass{peakNr},EPSXX_1,   '\epsilon _{xx}',refrow);set(gca,'FontSize',13)

%% Plot 3D bragg peak
%close all
%close('555')
for scans = 1:length(scans_incl)
    scans = scans_incl(scans);
    figure(555);   % plot 3D
    hold on
    scatter3(XYZr(scans,1,indices(scans,:)),XYZr(scans,2,indices(scans,:)),XYZr(scans,3,indices(scans,:)),60,I1D(indices(scans,:),scans),'fill','s'); hold on; axis equal; %set(gcf, 'MarkerFaceAlpha', 0.4);
    xlabel('x');ylabel('y');zlabel('z');
    drawnow;
end
hold off


%% Some variables and small calculations
%close Figure 23
%EPSXX_1 = ( 1.7142162./(COMvert)-1).*100;
%Wurtzite ref ->
EPSXX_1 = (1.7199464./(COMvert)-1).*100;
%Substrate ref->
%EPSXX_1 = (1.723776./(COMvert)-1).*100;
counter = 0;
%figure(24); clf;
%figure(25); clf;

```

```
InSb = 6.479;
aCub = 6.0583;


latticea = ((EPSXX_1./100)+1).*aCub;
xcomp = (latticea-aCub)./(InSb-aCub);

refrow = 35;


    % figure;
    % subplot(1,3,1);
    % %imagesc([1:15].*300/16,[1:47].*4/70,latticea(1:47,:)); colorbar; ; colormap('jet'); title('Lattice constant'); ylabel('\mu m'); xlabel('nm')
    % P3_plot_weighted(Mass(:,:,scan_Nr),latticea(:,:,scan_Nr),'Lattice constant a', refrow); %set(gca,'YDir','reverse')
    % subplot(1,3,2);
    % %imagesc([1:15].*300/16,[1:71].*4/70,xcomp(1:47,:)); title('x, InAs_{(x-1)}Sb_{x}'); colorbar; colormap('jet'); ylabel('\mu m'); xlabel('nm')
    % P3_plot_weighted(Mass(:,:,scan_Nr),xcomp(:,:,scan_Nr),'x, InAs_{(x-1)}Sb_{x}', refrow)



    % rowmean_x = nanmean(flipud(xcompfiltered(15:47,:)),2)
    %  x1 = nanmean(rowmean_x(1:6))
    %  x2 = nanmean(rowmean_x(7:10))
    %  x3 = nanmean(rowmean_x(11:17))
    %  x4 = nanmean(rowmean_x(18:end))

    % figure(324)
    % plot([15:47].*4/70,rowmean_x); title('X(l)'); xlabel('microns');




%% interpolate and plot x and mass for all scans

for scans = 1:length(scans_incl)
    scans = scans_incl(scans)
    % Normalizing each row of each scan
    for i = 1:71;
        mass_norm(i,:,scans) = Mass(i,:,scans)./nansum(Mass(i,:,scans));
        %mass_norm(mass_norm(i,:,scans)<2*mean(mass_norm(i,:,scans))) = 0;
        %xcomp(i,:,scans) = xcompnonorm(i,:,scans)./nansum(xcompnonorm(i,:,scans));
    end
    %Normalizing scans
    %mass_norm(:,:,scans) = mass_norm(:,:,scans)./nansum(nansum(mass_norm(:,:,scans)));
end

mass_norm(mass_norm<(1.1*mean(mass_norm(1,:,4))))=0;
%mass_norm(mass_norm>(1.3*mean(mass_norm(1,:,4))))=1;


for scan_Nr = 1:length(scans_incl)
    counter = counter + 1;
    scan_Nr = scans_incl(scan_Nr)


    clear interferometer_y interferometer30 interferometer60 xses yses
```

```
328     load(fullfile(vectorpath,[scanlist.textdata{scan_Nr+1}],strcat(sample_name,'.mat
            ')),'xses','yses','interferometer_y','interferometer30','interferometer60');
             % load each vector

330     Mass_int(:,:,scan_Nr) = interferometer(mass_norm(:,:,scan_Nr),interferometer_y,
            interferometer30,interferometer60,xses,yses,1);
        a_int(:,:,scan_Nr) = interferometer(latticea(:,:,scan_Nr),interferometer_y,
            interferometer30,interferometer60,xses,yses,1);
332  %   Mass_int= medfilt2(Mass_int(:,:,scan_Nr),[3,3]);
        %Mass_int(Mass_int<=0)=0;
334     [xcomp_int(:,:,scan_Nr),xgrid(:,:,scan_Nr),ygrid(:,:,scan_Nr)] = interferometer(
            xcomp(:,:,scan_Nr),interferometer_y,interferometer30,interferometer60,xses,
            yses,1);
        y(:,scan_Nr)=xgrid(:,1,scan_Nr)';
336     x(:,scan_Nr)=ygrid(1,:,scan_Nr)';


338
%     COMvert_int(:,:,scan_Nr) = interferometer(COMvert(:,:,scan_Nr),interferometer_y
    ,interferometer30,interferometer60,xses,yses,1);
340 %     xcomp_int = medfilt2(xcomp_int(:,:,scan_Nr),[3,3])
%       figure(23);
342 %       subplot(2,6,counter);
%       %P3_plot_weighted(Mass_int(4:end,3:end-2),xcomp_int(4:end,3:end-2),'x, InAs_{(
    x-1)}Sb_{x}',refrow)
344 %       imagesc(ygrid(1,:,scan_Nr),xgrid(:,1,scan_Nr),xcomp_int(:,:,scan_Nr)); daspect
    ([1,1,1]); colorbar; colormap('jet'); ylabel('\mu m'); xlabel('\mu m'); title('
    InAs(1-x)Sbx');
%       subplot(1,3,2);
346 %       imagesc(ygrid(1,:),xgrid(:,1),log(Mass(:,:,scan_Nr))); colorbar; daspect
    ([1,1,1]);%(15:46,7:end)); colorbar; %dascpect('[1,1,1]');
%       subplot(1,3,3);
348 %       imagesc(ygrid(1,:),xgrid(:,1),log(Mass_int)); colorbar; daspect([1,1,1]);
%       %
350 %       figure(24);
%       subplot(2,6,counter);
352 %       %P3_plot_weighted(Mass_int(4:end,3:end-2),xcomp_int(4:end,3:end-2),'x, InAs_{(
    x-1)}Sb_{x}',refrow)
%       P3_plot_weighted(Mass_int(3:end,4:end-4,scan_Nr),Mass_int(3:end,4:end-4,
    scan_Nr),'Intensity',refrow);
354 % %
% %       figure(23);
356 % %       subplot(2,6,counter);
% %       %P3_plot_weighted(Mass_int(4:end,3:end-2),xcomp_int(4:end,3:end-2),'x, InAs_
    {(x-1)}Sb_{x}',refrow)
358 % %       imagesc(COMvert_int(:,:,scan_Nr)); daspect([1,1,1]); colorbar; colormap('jet
    '); ylabel('\mu m'); xlabel('\mu m'); title('COMvert');
%
360 %
%       figure(25);
362 %       subplot(2,6,counter);
%       P3_plot_weighted(Mass_int(:,:,scan_Nr),xcomp_int(:,:,scan_Nr),'InAs(1-x)Sbx',
    refrow);
364


366 end

368 % %% Normalizing intensity
% for scans = 1:length(scans_incl)
370 %       scans = scans_incl(scans)
%       for i = 1:71;
372 %           mass_norm(i,:,scans) = Mass_int(i,:,scans)./nansum(Mass_int(i,:,scans));
%       end
```

```
%     %Normalizing scans
%     mass_norm(:,:,scans) = mass_norm(:,:,scans)./nansum(nansum(mass_norm(:,:,scans
    )));
% end


%% Aligning
%close all;

clear xcompalign massalign mass_all aalign mass_normalign
nrows = 74;
massalign = zeros(nrows,Realsize(2),length(scans_incl));
xcompalign = zeros(nrows,Realsize(2),length(scans_incl));
aalign = zeros(nrows,Realsize(2),length(scans_incl));
%alignmentrows = [49 47 54 54 52 51 51 50 49 56 62 69];
%alignmentrows = [53 54 47 52 46 51 56 63];
alignmentrows = [53 53 52 51 56 63]+10;
%alignmentrows = [73 72 72 72 72 71 70 71 74 81];
%alignmentrows = [72 72 74 74 73 72 70 72 73 80];

black = 1;

%massalign
refrow = 70;
%close('26')

for i = 1:length(scans_incl);
        xcompalign(nrows-alignmentrows(i):(nrows-alignmentrows(i))+70,:,i) =
            xcomp_int(:,:,scans_incl(i));
        massalign(nrows-alignmentrows(i):(nrows-alignmentrows(i))+70,:,i) = Mass_int
            (:,:,scans_incl(i));
        aalign(nrows-alignmentrows(i):(nrows-alignmentrows(i))+70,:,i) = a_int(:,:,
            scans_incl(i));

        figure(26)
        subplot(2,6,i)
        P3_plot_weighted(massalign(:,:,i),xcompalign(:,:,i),'InAs(1-x)Sbx',refrow,
            black);
        subplot(2,6,6+i)
        %P3_plot_weighted(massalign(:,:,i),massalign(:,:,i),'Intensity weighted',
            refrow,black);
        %subplot(3,10,20+i)
        imagesc(massalign(:,:,i)); colorbar; daspect([1,1,1]); title('Log(Intensity)
            '); colormap('jet')
end

%%
mass_all = nansum(massalign,3)./length(scans_incl);
xcomp_all = nansum(xcompalign.*massalign,3)./(mass_all.*length(scans_incl));
a_all = nansum(aalign.*massalign,3)./(mass_all.*length(scans_incl));

% for i = 77:84;
%     if mass_all(i,:) == 0
%         a_all(i,:) = aCub;
%         xcomp_all(i,:) = 0;
%     elseif mass_all(i,:) == NaN;
%         a_all(i,:) = aCub;
%         xcomp_all(i,:) = 0;
%     end
% end

%a_all(a_all(80:84,:)==0) = aCub;
sub_indices = isnan(a_all);
```

```
sub_indices(1:79,:)=0;


%
% a_all(sub_indices) = random('Normal',aCub,0.005,[78,1]);
% xcomp_all(sub_indices) = random('Normal',0,0.005,[78,1]);
% mass_all(sub_indices) = 0.02;
% mass_all(mass_all(77:84,:)==0) = nanmean(nanmean(mass_all(77,:)));
% mass_all(isnan(mass_all(77:84,:))) =nanmean(nanmean(mass_all(77,:)));

black = 1;

figure(72)
subplot(1,2,1)
aweighted = P3_plot_weighted(mass_all(10:end,:),a_all(10:end,:),'Lattice constant, a
    ',refrow,black);
subplot(1,2,2)
xcompweighted = P3_plot_weighted(mass_all(10:end,:),xcomp_all(10:end,:),'InAs(1-x)
    Sbx',refrow,black);

%P3_plot_weighted(mass_all,a_all,'Lattice Constant, a', refrow,black)
% %
% figure(75)
% subplot(1,2,1)
% imagesc(xcomp_all(2:end,:)); colorbar; colormap('jet');
% subplot(1,2,2)
% imagesc(a_all(2:end,:)); colorbar; colormap('jet');


figure(74)
plot(nanmean(flipud(xcompweighted(5:end-21,3:8)),2),[1:49])

%% Strain calculations of different sections
meansub = nanmean(nanmean(aweighted(74:66,:)));
mean0 = nanmean(nanmean(aweighted(54:65,:)));
mean1 = nanmean(nanmean(aweighted(49:53,:)));
mean2 = nanmean(nanmean(aweighted(44:49,:)));
mean3 = nanmean(nanmean(aweighted(36:43,:)));
mean4 = nanmean(nanmean(aweighted(11:35,:)));
mean5 = nanmean(nanmean(aweighted(1:10,:)));

strain = zeros(size(aweighted));

strain(74:66,:) = (aweighted(74:66,:)-meansub)/(meansub)*100;
strain(54:65,:) = (aweighted(54:65,:)-mean0)/(mean0)*100;
strain(49:53,:) = (aweighted(49:53,:)-mean1)/(mean1)*100;
strain(44:49,:) = (aweighted(44:49,:)-mean2)/(mean2)*100;
strain(36:43,:) = (aweighted(36:43,:)-mean3)/(mean3)*100;
strain(11:35,:) = (aweighted(11:35,:)-mean4)/(mean4)*100;
strain(1:10,:) = (aweighted(1:10,:)-mean5)/(mean5)*100;

figure(75)
P3_plot_weighted(mass_all(10:end,:),strain,'Strain',refrow,black);

%Zoom on top part:
refrow = 10;
figure(76)
P3_plot_weighted(mass_all(21:45,1:18),strain(11:35,1:18),'Strain',refrow,black);
```

E Matlab Code: Interferometer correction

```
function [interpolation,xgrid,ygrid] = interferometer(interpolant,int,int30,int60,
    x_pos,y_pos,snakescan)
int=reshape(int,length(y_pos),length(x_pos));
int=int./1000;

int30 = reshape(int30,length(y_pos),length(x_pos))./1000000;
int60 = reshape(int60,length(y_pos),length(x_pos))./1000000;

if snakescan
    for even = 2:2:length(int)
        int(even,:) = fliplr(int(even,:));
        int30(even,:) = fliplr(int30(even,:));
        int60(even,:) = fliplr(int60(even,:));
    end
end

int_x30 = int30.*sqrt(3)/2; %cos(30deg)=sqrt(3)/2
int_y30 = int30.*sqrt(1)/2; %sin(30) = 1/2

int_x60 = int60.*(sqrt(1)/2);%(1/2);%(1/2);
int_y60 = int60.*(sqrt(3)/2);
%

intx = (int_x30+int_x60)./2;
int = (int+int_y30+int_y60)./3;

%intx = intx./1000;

%int_x=(int_x30+int_x60)./2;
int_x = repmat(y_pos,[1,length(x_pos(:,1))])-intx;
int_y = repmat(x_pos,[1,length(y_pos(:,1))])'-int;

ygridorig=min(min(int_x)):((max(max(int_x))-min(min(int_x)))/(length(y_pos)-1)):max(
    max(int_x));
xgridorig=min(min(int_y)):((max(max(int_y))-min(min(int_y)))/(length(x_pos)-1)):max(
    max(int_y));

% xgridvec=mean(min(int_x)):((mean(max(int_x))-mean(min(int_x))))/(length(y_pos)-1):
    mean(max(int_x));
% ygridvec=mean(min(int)):((mean(max(int)))-(mean(min(int))))/(length(x_pos)-1):mean
    (max(int));

[ygrid xgrid] = meshgrid(xgridorig , ygridorig);
%[xgrid ygrid] = meshgrid(xgridvec, ygridvec);

% plot(xgrid(:),ygrid(:),'.')

interpolant=scatteredInterpolant(int_x(:),int_y(:),interpolant(:),'natural','none');
    %,'none');
interpolation = interpolant(xgrid,ygrid);
%interpolation = interpolation'
end


% %% Plotting
% close all
% colormap(hot)
% %set(gcf,'renderer','opengl');
% %hold off;
```

```
% threshold=10;
% subplot(2,2,1)
% %imagesc(flipud(ygrid(:,1)),flipud(xgrid(1,:)),fluor_interpolation')
% P3_plot_weighted(fluor_interpolation',fluor_interpolation','Fluor Interpolated',
    threshold);
% subplot(2,2,2)
% %imagesc(flipud(ygrid(:,1)),flipud(xgrid(1,:)),mass_interpolation')
% P3_plot_weighted(mass_interpolation',mass_interpolation','Diffraction',threshold);
% subplot(2,2,3)
% %imagesc(flipud(ygrid(:,1)),flipud(xgrid(1,:)),COMx')
% P3_plot_weighted(mass_interpolation',COMx','COMx',threshold);
% subplot(2,2,4)
% %imagesc(flipud(ygrid(:,1)),flipud(xgrid(1,:)),COMy')
% P3_plot_weighted(mass_interpolation',COMy','COMy',threshold)
```

F Matlab Code: Plotting, making low intensities black

Modified script, original provided by Tomas Stankevic.

```
function [I] = P3_plot_weighted (I0,I,name,refrow,black)
cla
Imask_max = max(max(I0(1:refrow,:))); maskthres = 0.2; maskgamma = 2;
Imask = imresize(I0,1,'bilinear'); %Imask = Imask+maskthres*Imask_max;
Imask = medfilt2(Imask,[3,3],'symmetric');
Imask(Imask>maskthres*Imask_max)=maskthres*Imask_max;
Imask = (Imask./max(Imask(:))).^maskgamma;
%drange = [min(I(Imask>0.99)),max(I(Imask>0.99))];
%set(gca,'Clim',[drange(1),drange(2)]);
I(Imask<0.1)=NaN;
imagesc(I); title(name,'FontSize',13); colorbar;
%set(gca,'YTickLabel',[]);
daspect([1,1,1]); colormap('jet');
if black
    black = cat(3, zeros(size(Imask)),zeros(size(Imask)), zeros(size(Imask)));
    hold on; freezeColors;
    h = imagesc(black); axis image; set(h, 'AlphaData', 1-Imask.*2);
end
end
```