Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Materials Science and Engineering

Olav Leth-Olsen

Distinguishing Martensite from Ferrite in Mild Steels with EBSD using Dictionary Indexing

Master's thesis in MTMT
Supervisor: Jarle Hjelen
Co-supervisor: Morten Karlsen
June 2023

NTNU
Norwegian University of
Science and Technology

Olav Leth-Olsen

Distinguishing Martensite from Ferrite in Mild Steels with EBSD using Dictionary Indexing

Master's thesis in MTMT
Supervisor: Jarle Hjelen
Co-supervisor: Morten Karlsen
June 2023

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Materials Science and Engineering

NTNU
Norwegian University of
Science and Technology

Preface

This master's thesis have been presented to the Department of Materials Science and Engineering at the Norwegian University of Science and Technology (NTNU) in the spring of 2023. It is a component of the TMT4905 Materials Technology Master's Thesis course. This project expand on a previous project carried out in the fall of 2022, which involved the establishment of the EBSP Indexer software.

I am appreciative of the chance to work on this project alongside my teammates, Hallvard Relling and Erlend Østvold. Their support and contributions were key, as was the team spirit they maintained. I am also thankful to my supervisor, Professor Jarle Hjelen, for his guidance and enthusiasm throughout the project, as well as to Professor Morten Karlsen, my co-supervisor, for his valuable feedback. PhD Candidate Håkon Wiik Aanes deserves special thanks for his assistance and excellent documentation on kikuchipy. I also want to acknowledge Senior Engineer Morten Peder Raanes, Senior Engineer Yingda Yu and Senior Engineer Sergey Khromov for their help with EPMA and EBSD analysis, Staff Engineer Berit Vinje Kramer and Staff Engineer Ruben Åge Hansen for their help with sample preparation, and Professor Jan Ketil Solberg for his useful feedback. Thank you to Cato Dybdal from SINTEF for providing the sample that was used in this thesis.

Abstract

Gaining insight into the structure and composition of materials is essential in the field of materials science. Electron backscatter diffraction (EBSD) in the scanning electron microscope (SEM) is a widely used characterization technique that allows for the analysis of the microstructure and crystallography of crystalline samples. The aim of this master's thesis is to explore the limits of an area of dictionary indexing (DI) by investigating the possibility to correctly identify martensite regions in a mild steel sample consisting of ferrite and martensite. The indexing was performed in EBSP Indexer, an open-source EBSD analysis software based on kikuchipy, developed for Mac and Windows. As a part of this project, EBSP Indexer was also further improved upon.

In this work, a 27MnCrB5-2 steel alloy was examined. Investigations showed ferrite bands within the martensite matrix. In mild steels, the difference in crystal structure between ferrite and martensite is very small, with martensite having a BCT crystal structure with a c/a relationship that is dependent on the carbon concentration, $c/a = 1 + 0.045 \times \text{wt\%C}$.

A collection of martensite master patterns were generated using EMsoft, with tetragonality based on carbon concentrations ranging from 0.1 to 1.5 wt%. Dictionary indexing was successively applied with different methods for pattern center optimization and orientation refinement. The pattern center parameter proved to be critical in determining the phases. By executing pattern center refinement in kikuchipy with a large number of high-quality ferritic patterns, the coordinates for the pattern center converged precisely, providing convincing indexing results.

The findings of this project illustrate that it is feasible to distinguish between ferrite and martensite in mild steels by use of EBSD dictionary indexing and precise indexing parameters, supported by results from EPMA. Given the crucial role of accurate pattern center optimization, the tool for pattern center optimization via pattern selection has been effectively integrated into the EBSP Indexer software. This allows scientists facing similar difficulties to easily select suitable EBSPs for pattern center calibration, consequently delivering more accurate EBSD results than previously possible.

Sammendrag

Å få innsikt i strukturen og sammensetningen av materialer er avgjørende innen materialvitenskap. Elektron-tilbakespredningsdiffraksjon (EBSD) i elektronmikroskopet (SEM) er en mye brukt karakteriseringsteknikk som muliggjør analyse av mikrostrukturen og krystallstrukturen til krystallinske prøver. Målet med denne masteroppgaven er å utforske mulighetene innenfor et område av ordboksindeksering (DI) ved å undersøke muligheten for å korrekt identifisere martensittområder i et lavkarbonstål bestående av ferritt og martensitt. Indekseringen ble utført i EBSP Indexer, en open-source programvare for EBSD analyse basert på kikuchipy, utviklet for Mac og Windows. EBSP Indexer har blitt utvidet som del av dette master-prosjektet.

I dette arbeidet ble en 27MnCrB5-2 stål-legering analysert. Undersøkelser viste ferrittbånd innenfor martensitt-matrisen. I lavkarbonstål er forskjellen i krystalstruktur mellom ferritt og martensitt beskjeden, med martensitt som har en BCT krystalstruktur med et c/a-forhold avhengig av karbonkonsentrasjonen, $c/a = 1 + 0.045 \times \text{wt\%C}$.

En rekke teoretiske referansemønstere ble generert ved bruk av EMsoft, med tetragonalitet basert på karbonkonsentrasjoner som spenner fra 0,1 til 1,5 wt%. DI ble suksessivt brukt med forskjellige metoder for optimalisering av projeksjonssenter og orienteringsoptimering. Projeksjonssenteret som ble brukt ved indeksering viste seg å være avgjørende for å bestemme korrekt fase. Ved å utføre optimalisering av projeksjonssenter i kikuchipy med et stort antall ferrittmønstre av høy kvalitet, konvergerte koordinatene for projeksjonssenteret presist, noe som førte til realistiske indekseringsresultater.

Resultatene av dette prosjektet illustrerer at det er gjennomførbart å skille mellom ferritt og martensitt ved bruk av EBSD ordboksindeksering og presise indekseringsparametere, underbygget av resultater fra EPMA. Gitt den avgjørende rollen av nøyaktig optimalisering av projeksjonssenter, har verktøyet for optimalisering av projeksjonssenter ved å velge EBSD mønstere blitt integrert i EBSP Indexer-programvaren. Dette lar brukere med lignende problemstillinger enkelt velge egnede EBSD mønstere til kalibreringen av projeksjonssenteret, og dermed forbedre EBSD resultatene.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Electron backscatter diffraction (EBSD) is a powerful tool for the microscopic study of crystalline materials. This technique relies on examining the diffraction patterns formed when a beam of electrons interacts with a crystallographic specimen, thereby enabling the determination of a material's crystal structures, grain orientations, phases, and deformation at a nanoscale level [1]. EBSD finds essential applications in fields such as materials science and geology, and requires a flat, polished specimen inclined at a specific angle within the scanning electron microscope (SEM).

The EBSD process involves scanning the specimen with an electron beam in a grid pattern, subsequently examining the diffraction patterns to interpret the crystallographic orientation and phase of each point. The two major indexing approaches include Hough indexing (HI) and dictionary indexing (DI) [1, 2]. While Hough indexing leverages the Hough transform for quicker results, it may lack precision. On the other hand, dictionary indexing requires more computational effort as it compares experimental patterns directly with a simulated master pattern. Despite this, it is more resilient to noise and lower resolution patterns, and can differentiate between phases of similar crystal structures.

This project proceeds to test what dictionary indexing is capable of, by distinguishing between martensite and ferrite in a mild steel sample. In mild steels, the contrast in the crystal structure between martensite and ferrite is marginal, with ferrite exhibiting a BCC crystal structure and martensite having a BCT crystal structure. The lattice parameters of martensite is linearly dependent of the carbon content, and can be described using the formula: $c/a = 1 + 0.045 \times \text{wt\%C}$ [3]. Given the similarity in their crystal structures, the master patterns, fundamental to dictionary indexing, are also remarkably similar. Therefore, achieving the desired outcome necessitates high-quality experimental data and highly accurate indexing parameters. Obtaining an accurate

pattern center (PC) emerged as the most challenging task.

A method for pattern center optimization evolved through a process of trial and error. By selecting numerous high-quality ferritic electron backscatter diffraction patterns (EBSPs), pattern center optimization in the open-source Python library for EBSD analysis kikuchipy [4] yielded an accurate PC, leading to highly convincing indexing results. The carbon profile acquired from the DI phase map displayed resemblance to the carbon profile obtained by the electron probe microanalysis of the same region of the sample.

Given the success of the pattern center technique, it has been incorporated into EBSP Indexer [5], the graphical user interface based on kikuchipy. The pattern center optimization tool enables users to conveniently select the EBSPs for PC optimization directly from the EBSD dataset, promoting a user-friendly experience. This makes dictionary indexing more accessible to users struggling with challenging indexing situations. EBSP Indexer, developed in collaboration with Hallvard Relling and Erlend Østvold, can be freely accessed at https://github.com/EBSP-Indexer/EBSP-Indexer.

Chapter 2

Theoretical Background

This chapter will present an introduction to the principles of EBSD and EBSD indexing, as well as ferritic and martensitic microstructure in carbon steels. Additionally, suggestions for further reading on these topics will be offered for those interested in exploring the current research landscape.

2.1 Electron Backscatter Diffraction

Electron backscatter diffraction (EBSD) is a powerful characterization method used to analyze polycrystalline materials [1]. It can identify phases and phase distribution, grain size, and crystallographic orientations. This technique relies on a scanning electron microscope (SEM) equipped with an EBSD detection system, and has become well-established in the fields of materials science and geology. Understanding the microstructure of a material is crucial for controlling and predicting its physical properties. EBSD is an effective tool for this purpose. Its high-speed data collection and the increasing availability of SEMs make it a popular choice for material analysis.

EBSD patterns (EBSPs) are generated through the backscattering of electrons from a sample in a vacuum chamber. An electron probe is used to accelerate electrons towards the specimen and scan a specific region of interest [1]. Typically, an acceleration voltage of 20 keV is used. The electrons interact with the surface atoms at an angle, usually 70°, and penetrate to a depth of approximately 20-80 nm. An illustration of a typical EBSD setup is given in Figure 2.1.

The electrons scatter randomly upon interacting with the atoms on the upper layer of the sample. However, due to the crystal structure, electrons are channeled in specific

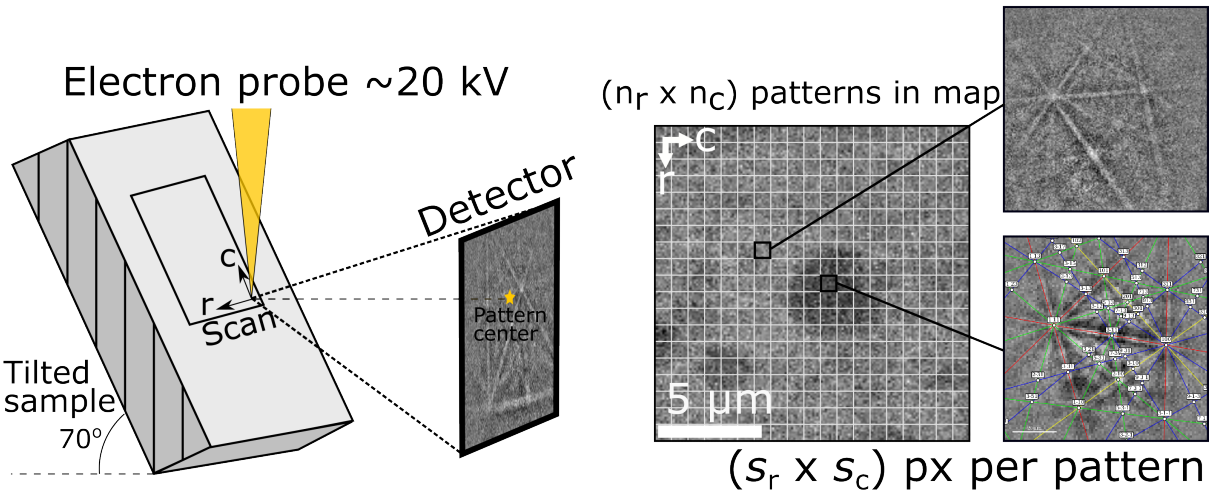


Figure 2.1: The EBSD configuration involves an electron beam interacting with a sample set at a tilt. This interaction generates backscattered electrons that form an EBSP on the detector. The pattern center, indicated by a star, is the point on the detector closest to the interaction region. On the right side of the illustration, the process of indexing patterns from the grid is depicted, which involves assigning orientations to the Kikuchi bands. Figure from [6].

directions before they hit the phosphor detector. Winkelmann provides a detailed description of this mechanism in "Many-beam dynamical simulation of electron backscatter diffraction patterns" [7]. These mechanics lead to the formation of patterns called backsatter Kikuchi patterns or electron backscatter diffraction patterns (EBSPs). The Kikuchi bands within these patterns represent crystallographic planes in the crystal lattice. The closer packed planes provide stronger and brighter backscattering signals. Bragg's formula, equation 2.1, is used to determine the width of the Kikuchi bands. d_{hkl} denotes the interplanar spacing, n represent the order of diffraction, λ stand for the wavelength of the electrons emitted by the electron beam, and θ_{hkl} represent the Bragg's angle.

$$2d_{hkl} \sin \theta_{hkl} = n\lambda \tag{2.1}$$

The electron beam moves in a grid pattern, collecting an EBSP at each grid coordinate. Modern EBSD systems can generate hundreds of patterns per second [1]. After the EBSP is formed on the phosphor detector, a CCD or CMOS sensor captures the light that is emitted [8]. By combining pixels on the sensor, a technique known as binning,

the sensitivity and speed of the system can be increased while reducing noise and improving contrast and transfer speeds. However, this can result in a loss of detail due to the lower resolution of the patterns.

Crystallographic planes in polycrystalline materials can be indexed using the backscatter Kikuchi patterns generated by EBSD [1]. These planes can be classified by their Miller index {hkl}, and the intersection of these planes corresponds to crystallographic directions [uvw] in the crystal. The angle between a pair of intersecting Kikuchi bands corresponds to the angle of the crystallographic planes from which they originate. Historically, EBSD indexing has primarily relied on Hough indexing (HI) as the primary indexing method (explained in 2.2). However, dictionary indexing, or DI (explained in section 2.3), has emerged as a promising alternative that offers distinct advantages [9], particularly in situations where Hough indexing fails to meet the necessary requirements.

Both Hough- and dictionary indexing require accurate parameters to provide precise results. Both methods analyze experimental data by comparing them with theoretical outcomes. HI utilizes peaks in Hough Space from a Hough transform, whereas DI compares the entire pattern to simulated patterns [1, 2]. For the theoretical data to align with the experimental data, the calibration settings of the experiment need to be in consistent with the parameters supplied to the generate the look-up table for HI and the master pattern projection for DI. These parameters include the pattern resolution, sample tilt, detector tilt, and pattern center (PC). Although the first three are straightforward and self-explanatory, the pattern center is relatively complex due to the difficulties in its precise measurement and the existence of multiple PC conventions.

2.1.1 Pattern Center

The pattern center (PC) is situated at the closest point on the detector to the specimen's interaction region, and is defined in relation to the screen coordinate system. It plays a crucial role in generating the master pattern projections (defined in section 2.3.1), and in calibrating look-up values for peaks in Hough space. The pattern center can be found in several ways, including the moving screen method [10], the use of a global search algorithm and averaging the results from multiple patterns [11], and methods provided by conventional software such as Bruker ESPRIT and EDAX/TSL OIM Data Collection.

The pattern center is characterized by a triplet of coordinates (x*, y*, z*), whose interpretations vary depending on the chosen pattern center convention. Both BRUKER and EDAX/TSL conventions base their measurements on the relative distance from the pattern's edge, in relation to the overall dimensions of the pattern. In the case of square patterns, both BRUKER and TSL conventions agree on x* and z*, yet differ on y*. Specifically, BRUKER convention determines y* from the top of the pattern, whereas TSL convention measures y* from the bottom. Consequently, y* values in BRUKER and TSL conventions are complementary to 1, meaning the sum of y* values from both conventions will always equal 1. Readers seeking further detail are encouraged to explore the kikuchipy User Guide [12] for a more comprehensive understanding.

2.2 Hough Indexing

Hough Indexing (HI) is a type of automated indexing method that uses the power of the Hough transform to identify the locations of Kikuchi bands [1]. The Hough transform is a computational procedure that detects straight lines within an image by transforming these lines into peak representations within a Hough space. Two key concepts must be explained to understand this transformation - ρ and θ, as used within image space. Here, ρ denotes the shortest distance from the origin to a given line, while θ represents the angle formed between the line created by ρ and the x-axis. This concept is shown graphically in Figure 2.2. The line in image space is indicated by the point (θ*, ρ*) in Hough space, where the sine waves intersect. Each sine wave is a Hough transform of one of the three points on the line in image space. The Hough transform is described by equation 2.2, where x and y denotes the coordinate in image space, and ρ and θ represents dependent and independent variable in Hough space, respectively.

$$\rho = x \cos\theta + y \sin\theta \tag{2.2}$$

When the Hough transform is applied to all pixels that make up an EBSP, the intensities of each pixel are combined, and peaks of intensity correspond to distinct Kikuchi bands. This is illustrated in Figure 2.3, where backscattered Kikuchi patterns going through this transformation produce a grey image with white peaks of intensity. These bright spots are identified and compared to look-up values for relevant phases and

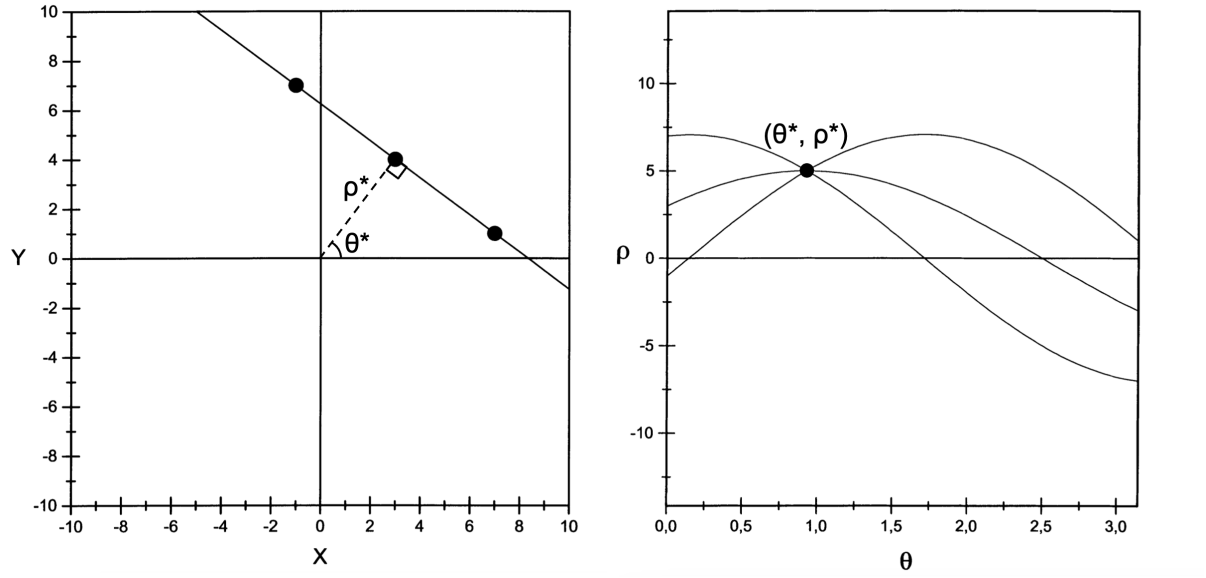


Figure 2.2: Hough transform of points on a line (left) to Hough space (right) [13].

orientations. The look-up values varies depending on indexing parameters such as the pattern center, sample tilt and detector tilt. The orientation and phase are assigned to the best fit.

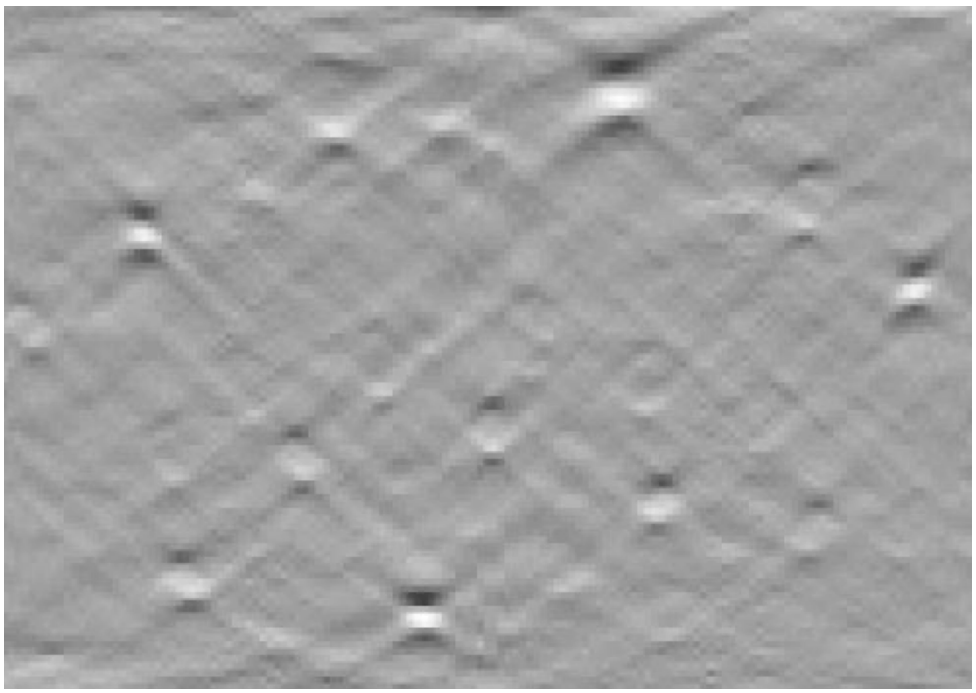


Figure 2.3: Hough transform of an EBSP. Bright points each represent a Kikuchi band.

2.3 Dictionary Indexing

Dictionary indexing (DI) is a pattern matching method that provides an alternative to Hough indexing for EBSD indexing. It was introduced in 2013 by De Graef and Callahan [2] and is now a viable option thanks to the increased processing power of personal computers. Unlike Hough indexing, DI compares experimental patterns to theoretical patterns pixel by pixel to find the best matching orientation and phase.

The key to DI is the master pattern, described in more detail in section 2.3.1. Master patterns are theoretical EBSD projections created using many-beam dynamical simulation. This creates all possible diffraction patterns around the crystal in a Kikuchi sphere. Only fractions of the sphere are needed, depending on the symmetry of the crystal structure, in order to identify all crystallographic orientations when indexing. For BCC crystal structures, only a 24th of the master pattern is necessary to generate all theoretical patterns, whereas for BCT crystal structures, which possess lower symmetry, an 8th is required.

The orientation space that the master pattern creates is continuous and spherical, but flat images are required for pattern matching. To make the images comparable to experimental patterns, parts of the master pattern are projected onto flat images using gnomonic projection [12]. A visualization of gnomonic projection can be seen in Figure 2.4.

While the original master pattern model is a continuous form, projecting the orientations using gnomonic principles generates a distinct, finite number of simulated models. A low angular step size between projections will deliver more precise matches and more accurate results, at the cost of a larger dictionary of simulated patterns and slower indexing speeds. In order to ensure a high degree of similarity between the projected patterns and the experimental patterns, it is important that the projection parameters align with the experimental parameters. The parameters that need to match include the pattern center, pattern pixel resolution, sample tilt, and detector tilt.

Each simulated pattern in our dictionary must be compared with every experimental pattern to identify the best alignment or match. To do this, a standard tool for measuring similarity called the zero-mean normalized cross-correlation (ZNCC) [15] is used. It is also known as the Pearson correlation [4].

This ZNCC tool gives a coefficient, or a numerical value (r), that represents how

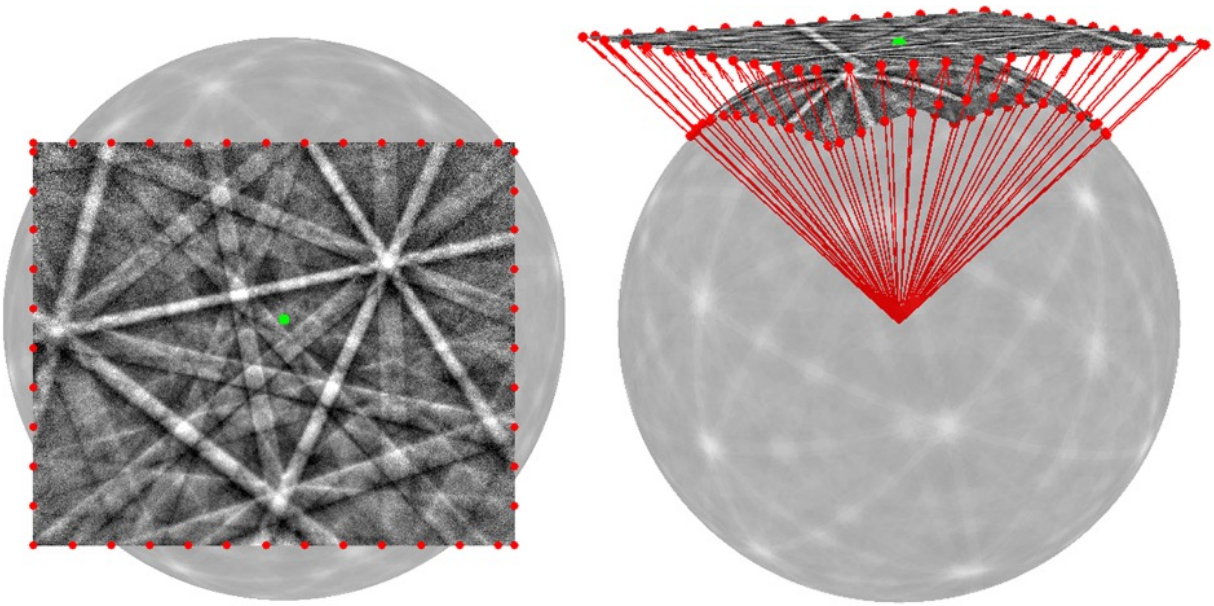


Figure 2.4: Gnomonic projection of a master pattern, from [14].

similar two patterns are. To calculate this value (r), Equation 2.3 is used, which involves subtracting the average pixel value from each pixel in the image and then comparing these adjusted pixel values between the experimental and simulated images. This is shown by Equation 2.3, where x represents the value of a pixel from the experimental pattern, y represents the corresponding pixel from the simulated pattern, and n represents the number of pixels. The most likely orientation is determined by comparing the similarity measure (r-value) for each simulated pattern with the experimental pattern, and selecting the one with the highest value. To determine the phase from which the pattern originates, the highest r-value from one phase is compared with the highest r-values from other phases. The phase with the highest r-value is considered to be the most probable source of the pattern. The r-value maxes out at 1 (perfect match) and the lowest value is -1 (perfectly opposite). According to Winkelmann et.al [16], r-values exceeding 0.6-0.7 are deemed visually convincing fits for dictionary indexing.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.3}$$

2.3.1 Master Pattern

The master pattern refers to a global simulated EBSD pattern. It represents the ideal representation of all EBSPs under the assumption of flawless crystals and the absence of any noise or interference from experimental equipment. This involves simulating a small spherical single crystal sample being targeted with electron beams from every direction, and computing the backscattering of electrons based on Bloch wave theory with a stochastic Monte Carlo simulation of the energy, depth, and directional distributions [7]. The procedure for generating master patterns can be carried out using the software EMsoft [17].

In practice, the process of creating master patterns involves several steps. First, crystallographic information such as the unit cell parameters and space group of the desired phase is imported into the software. This information can be obtained from experimental data or from crystallographic databases such as *Pearson's Handbook of Crystallographic Data for Intermetallic Phases* [18]. Next, a virtual crystal is created by specifying the crystallographic parameters. The virtual crystal represents the phase being studied.

Using the virtual crystal, EMSoft then simulates diffraction patterns that would be obtained from the material using an electron beam by use of Monte Carlo simulation. Monte Carlo simulations work by defining a model, the virtual crystal, then repeatedly sampling electrons according to a probability distribution and calculating the corresponding output [19]. By repeating this process many times, it is possible to build up a statistical picture of the range and likelihood of outcomes. After gathering the data from the simulated backscattering of electrons, the master pattern may be constructed.

The master pattern serves as a complete representation of all possible diffraction patterns, a Kikuchi sphere as perceived by a spherical detector surrounding the virtual crystal. The master pattern then functions as a standard when compared to experimental diffraction patterns obtained from the investigated material. The comparison can help identify crystallographic orientations and phases that are present in the material.

2.3.2 Pattern Processing

Prior to performing indexing of an EBSD dataset, it is essential to correct the pattern background from systematic errors, such as shading and artifacts on the detector. A high signal-to-noise ratio is important for improving indexing results. Achieving this

involves subtraction of static and dynamic background. Averaging of neighbouring patterns may be applied to remove noise further. The pattern processing methods utilized by kikuchipy [4] and EBSP Indexer align with the procedures described here.

The static background refers to an intensity distribution or signal that does not carry direct information about the crystallographic orientation or phase of the material but adds a consistent shaded layer to the diffraction image [4]. The intensity of backscattered electrons on the detector decreases relative to the distance from the pattern center. This results in a pattern that appears darker around the edges. Additionally, any persistent damage on the detector affects all acquired patterns consistently. Such effects can be removed through the elimination of the static background. The static background can be obtained by averaging the intensity of patterns with different orientations over an extensive area.

The inconsistency in pattern intensity, resulting from topographical variations (pores, etc.) across the scan area, can be normalized through the subtraction of the dynamic background. The dynamic pattern is derived from a process known as Gaussian blurring [15]. In this procedure, each pattern is individually blurred using a Gaussian window, the spread of which is determined by the standard deviation parameter. Following the application of dynamic background correction, each pattern should acquire a similar average intensity.

The signal-to-noise ratio in a pattern can be further improved by implementing an averaging process with similar neighbouring patterns. The technique of neighbor pattern averaging effectively enlarges the virtual interaction volume between the electron beam and the sample, which could potentially compromise the spatial resolution. Furthermore, in certain scenarios such as at grain boundaries, this averaging process may mistakenly combine two or more diffraction patterns from different grains, a situation which might not be desirable [20].

2.4 Software Tools

This section aims to provide readers with an overview of the software tools EMsoft, kikuchipy, and EBSP Indexer, and their applications in the field of materials science. The focus will be on discussing the capabilities of these tools, including their features for simulating, processing, and analysing EBSD data.

2.4.1 EMsoft

The open-source software EMSoft is primarily coded in Fortran-90, designed for calculating and visualizing scanning electron microscopy diffraction patterns [17]. These include EBSD, ECP, TKD, and EKP. The software package also includes programs for TEM defect image contrast, CBED, PED, Laue x-ray diffraction, and a series of programs for computational polarized light microscopy. Each of these can be used independently via command-line and generates `HDF5` output files. One of the key features of EMSoft is its ability to generate master patterns. EMsoft also includes dictionary indexing tools.

2.4.2 kikuchipy

kikuchipy is an open-source Python-based library designed for the processing, simulation, and analysis of EBSD data. The central feature of Kikuchipy is dictionary indexing. The multi-dimensional data analysis is offered by the open-source HyperSpy library [21] and the crystallographic orientation and symmetry analysis is provided by orix [22]. kikuchipy also supports Hough indexing by wrapping the Python library PyEBSDindex [23].

kikuchipy offers utilities for optimizing pattern centers, visualization techniques, geometric simulation, and additional valuable tools applicable in the EBSD indexing field. The iterations of kikuchipy utilized in this project were variations of version 0.8. Currently, the software is at version 0.8.5 and undergoes regular updates by Håkon W. Ånes and the team of kikuchipy developers.

2.4.3 EBSP Indexer

EBSP Indexer is a dynamic, open-source software package crafted by a team of master students from the Norwegian University of Science and Technology (NTNU) [5]. This package, designed for the analysis of Electron Backscatter Diffraction data, presents a comprehensive selection of tools for processing, visualizing, and examination of EBSD data. It was developed using PySide6, which provides Python bindings for the Qt application framework [24], and heavily relies on kikuchipy [4] as its primary Python package.

The appealing and user-friendly interface of EBSP Indexer adds another level of accessibility for users, enabling straightforward navigation and execution of complex analysis tasks. The software features a variety of tools for loading and pre-processing EBSD data, including but not limited to, background correction, pixel binning, and visualization techniques such as image quality (IQ) maps and average dot product (ADP) map. EBSP Indexer incorporates both Hough and dictionary indexing for identifying the crystallographic orientation and phases of a material. In-depth exploration and discussion of EBSP Indexer can be found in the master's theses by Hallvard Relling and Erlend Østvold, titled "EBSP Indexer - using an open-source dictionary indexing software for phase differentiation" and "EBSP Indexer - An open-source alternative to commercial EBSD software", respectively [25, 26].

2.5 Phase Transformations in Hardening Steel

This section details the process of heat treatment aimed at achieving ferritic and martensitic micro structures in mild steels, using phase and transformation diagrams. The lattice properties of each phase is highlighted due to its importance when discussing EBSD master patterns and dictionary indexing.

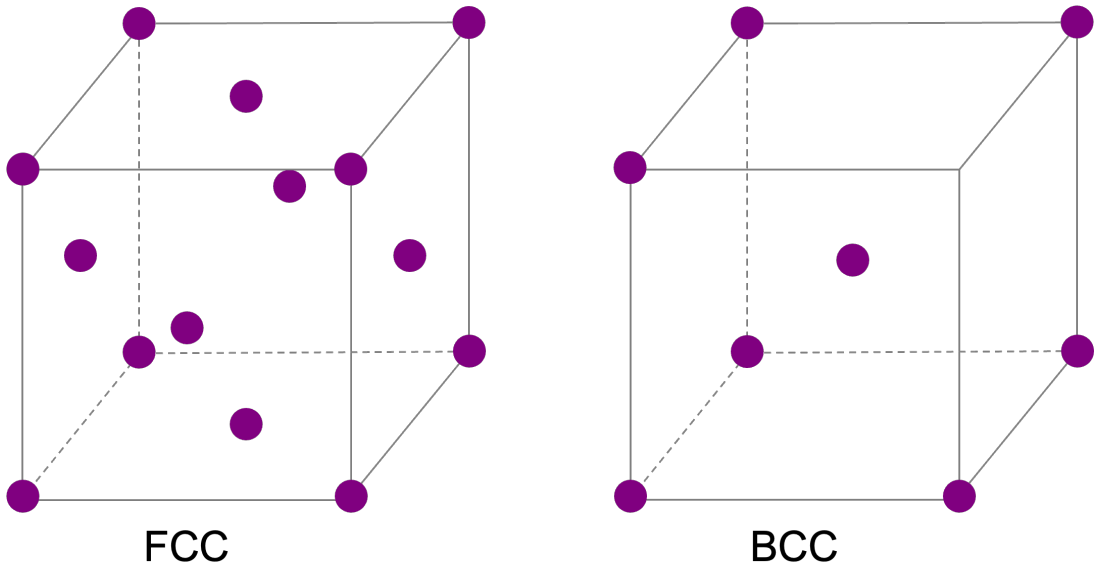


Figure 2.5: FCC and BCC unit cells, corresponding to the crystal structure of austenite and ferrite, respectively.

2.5.1 The Iron-Carbon Phase Diagram

To initiate an investigation into the characteristics of steel, a fundamental understanding of the iron-carbon phase diagram is essential. The diagram, as depicted in Figure 2.6, illustrates the equilibrium phases in steels along with their corresponding transformation temperatures. It is important to note that graphite, not cementite (Fe_3C), is the true equilibrium phase of carbon. The transformation time for graphite is too lengthy for practical purposes, providing cementite its place at the right in the phase diagram. The properties and transformation points of the phases can be altered through the incorporation of alloying elements.

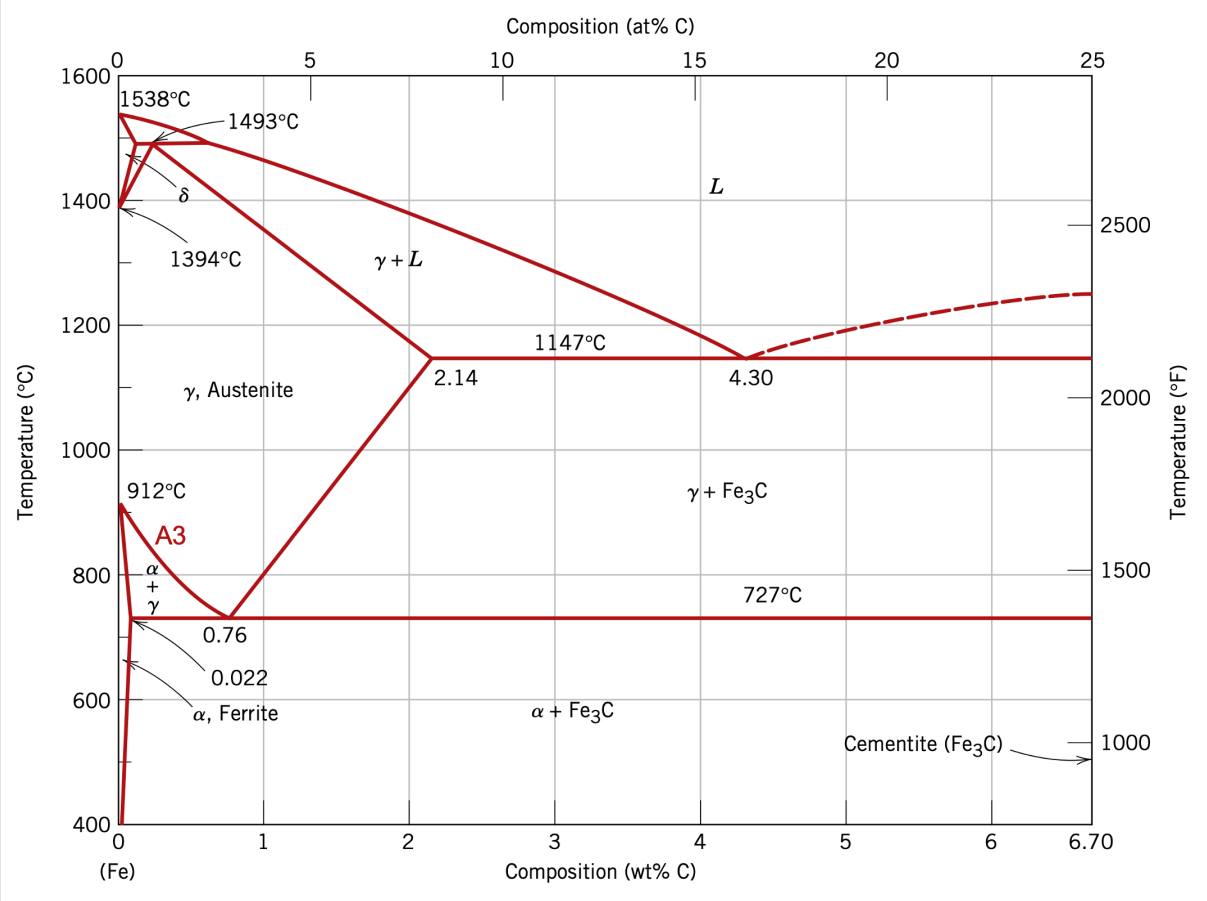


Figure 2.6: The Fe-Fe_3C phase diagram. Reprint of Figure 9.24 in Callister Materials Science and Engineering [27].

For a steel composition containing 0.25 wt% C, the austenitizing temperature is approximately 850°C, indicated by *A3* on Figure 2.6. Austenite has a face centred cubic

(FCC) structure, while ferrite a body centred cubic (BCC) structure, illustrated by Figure 2.5

2.5.2 Formation of Ferrite and Pearlite

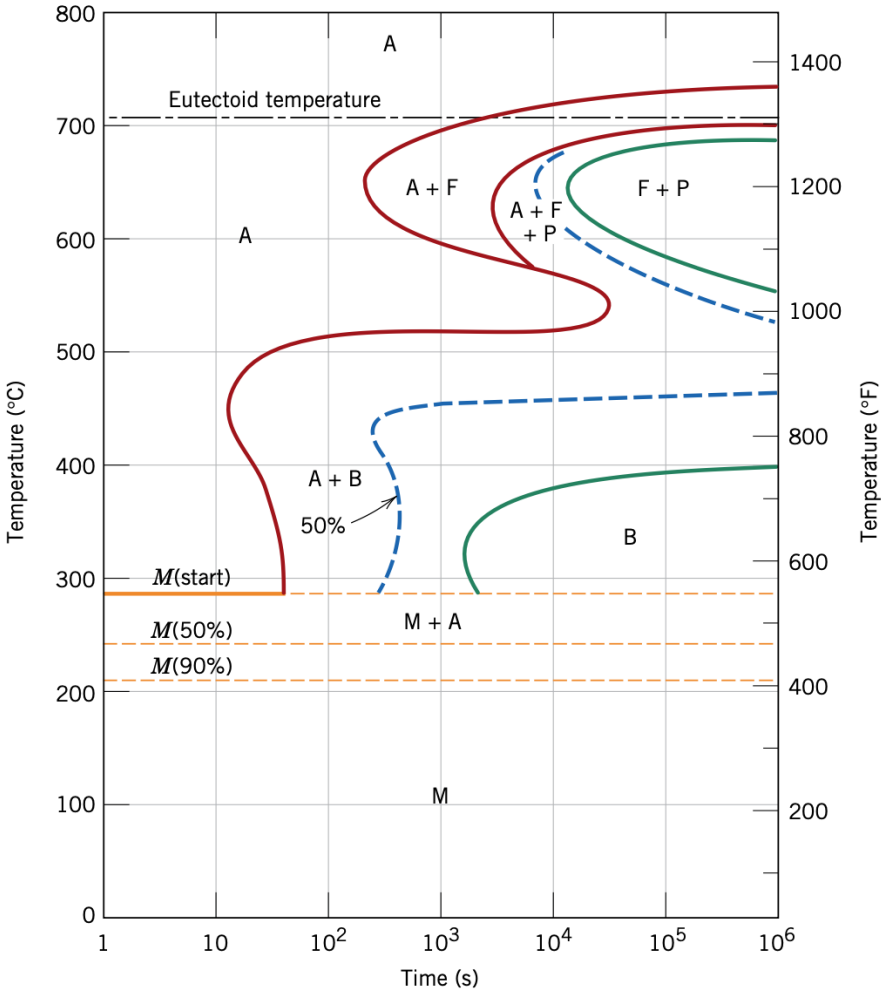


Figure 2.7: The figure presents an Isothermal Transformation (TTT) diagram for hypoeutectoid 4340 steel [27]. It depicts the transformations between different phases; austenite (A), ferrite (F), pearlite (P), bainite (B), and martensite (M), based on cooling time and temperature.

Although phase diagrams are valuable tools for understanding materials, they have a significant limitation in that they do not account for the kinetics of phase transformations. Time-temperature-transformation (TTT) diagrams are utilized to predict the

expected phase, as shown in Figure 2.7. The formation of specific phases is dependent upon both temperature and the driving force of the reaction. These parameters show an inverse relationship with one another. In other words, as the temperature drops, the reaction's driving force goes up. At the same time, lower temperatures also slow down diffusion [27].

Ferrite and pearlite formation happens when carbon diffusion is sufficient when cooling from the austenite region. Below A3, indicated on Figure 2.6, ferrite formation begins. Once the temperature hits eutectic temperature, 727°C, ferrite and cementite creates the eutectic phase pearlite. This is shown in the TTT diagram on Figure 2.7 as the "F+P"-region.

2.5.3 Formation of Martensite

Quenching of steel is deployed to produce martensite, and this process requires heating of the steel to a temperature at which austenite forms. To form martensite, the steel must be rapidly cooled from the austenite region, as shown in figure 2.7. Martensite is the hardest and most brittle phase of steel, due to the combination of supersaturated interstitial carbon and the large number of dislocations resulting from shear deformation [27].

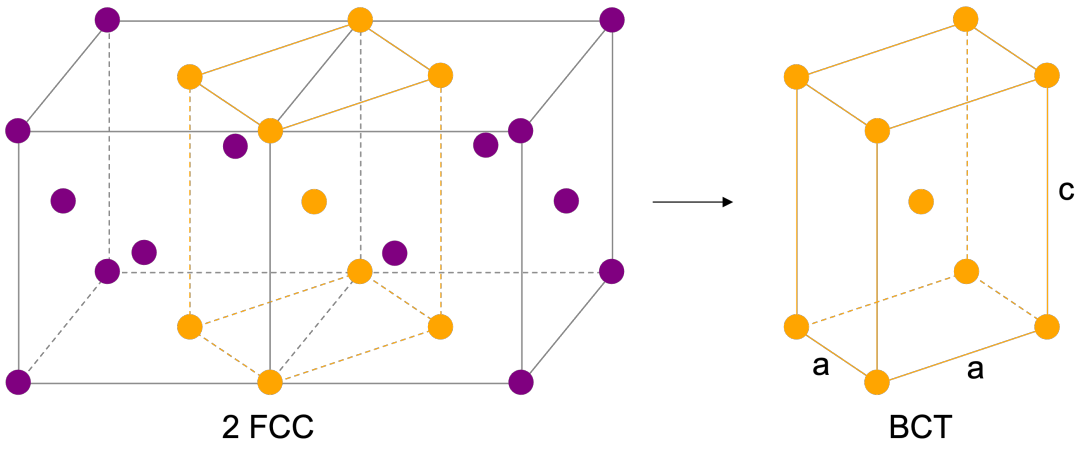


Figure 2.8: Austenite (FCC) to martensite (BCT) transformation in rapid cooling carbon steels. Higher carbon content corresponds to larger c/a relationship. Lattice parameters *a* and *c* are indicated for martensite.

When a steel is cooled below its critical temperature, it does not immediately form a particular phase, such as ferrite or pearlite. Martensite is a phase that forms at high

cooling rates, without the occurrence of diffusion. Fast cooling prevents carbon diffusion and traps it within the austenite matrix, which remains unstable. In this scenario, austenite undergoes transformation through shear deformation, resulting in the elongation of the face-centered cubic (FCC) structure to a body-centered tetragonal (BCT) structure, shown in Figure 2.8. The tetragonality of the martensite is dependent on the carbon concentration [3], described by Equation 2.4.

$$\begin{aligned} a &= 0.2861 + 0.0116 \times wt\%C \\ c &= 0.2861 - 0.0013 \times wt\%C \\ c/a &= 1.000 + 0.045 \times wt\%C \end{aligned} \tag{2.4}$$

Chapter 3

Experimental

This chapter provides a detailed overview of the steel alloy utilized in this research, as it was received, along with its chemical composition. The chapter then goes on to present the various processes implemented throughout the study, including the heat treatment applied to achieve distinct steel phases, preparation of the sample, and the configuration for EBSD analysis. Lastly, it thoroughly explains the operations conducted using the EBSD indexing software. EMsoft was used for generating master patterns, and EBSPIndexer and kikuchipy for pattern center calibration, dictionary indexing and refinement.

3.1 Manufacturing

Microstructural analysis of 27MnCrB5-2 after the hardening cycle reveals a ferrite-pearlite structure obtained through hot rolling of cast steel blocks followed by controlled temperature cooling. The steel composition is given in table 3.1.

Table 3.1: Chemical composition of the mild steel sample.

Element	C	Si	Mn	Cr	B
Composition (wt%)	0.25-0.29	0.15-0.35	1.15-1.35	0.30-0.50	0.002-0.006

During production at Raufoss, machined parts with thin walls were subjected to induction heating up to 900°C in 90 seconds, followed by rapid cooling in a water-based polymer under rotation and flushing to achieve uniform cooling. The cooling time to ambient temperature is reported in the literature to be beneficial for low Mn segregation. Process, steel, and wall thickness are qualified to produce martensite.

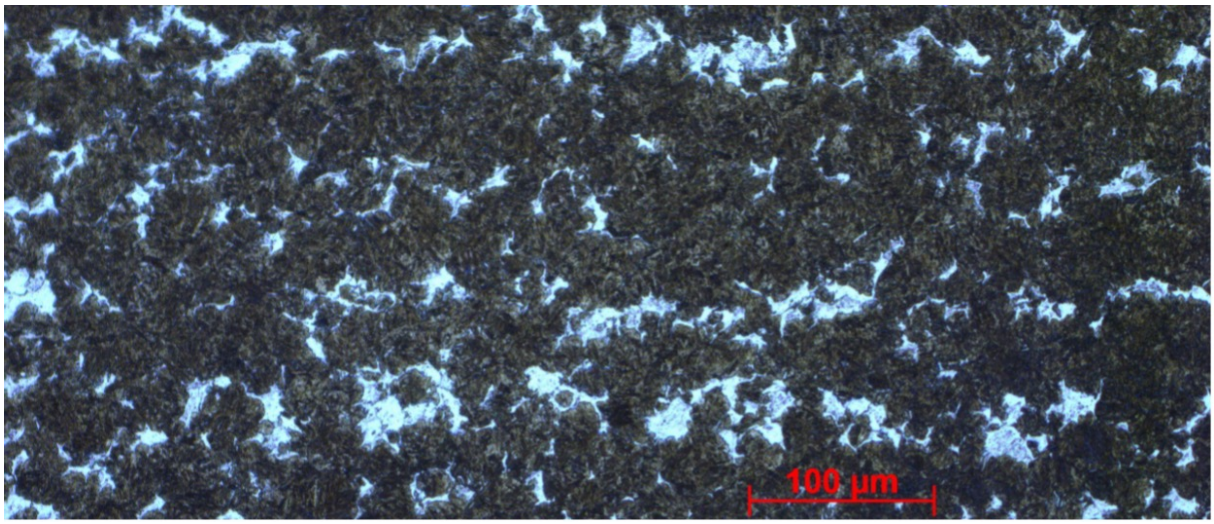


Figure 3.1: Steel structure observed in the primary metallurgical direction.

The examined steel, composed of ferrite and martensite, did not exceed the A3 phase boundary. A furnace change was necessary. The microstructure can be observed in the primary metallurgical direction in Figure 3.1. The primary metallurgical orientation is the longitudinal orientation of both the sample and the final product, which aligns with the rolling direction. This perspective facilitates the identification of macro/micro-segregation effects, which manifest as striped patterns in the longitudinal direction.

3.2 Specimen Preparation for EBSD

After cutting, the steel specimen was embedded in Struers' PolyFast resin. Subsequently, they underwent a series of preparation steps using the Tegrapol-31 from Struers:

1. Grinding with 220 SiC paper for 3 minutes.
2. Polishing with an Allegran/Allegro disk using a 9-micrometer Allegro/largo diamond suspension from Struers for 4 minutes.
3. Polishing with a Dac disk using a 3-micrometer MolR3 diamond suspension from Struers for 4 minutes.
4. Polishing with a Nap disk using a 1-micrometer NapR1 diamond suspension from Struers for 4 minutes.

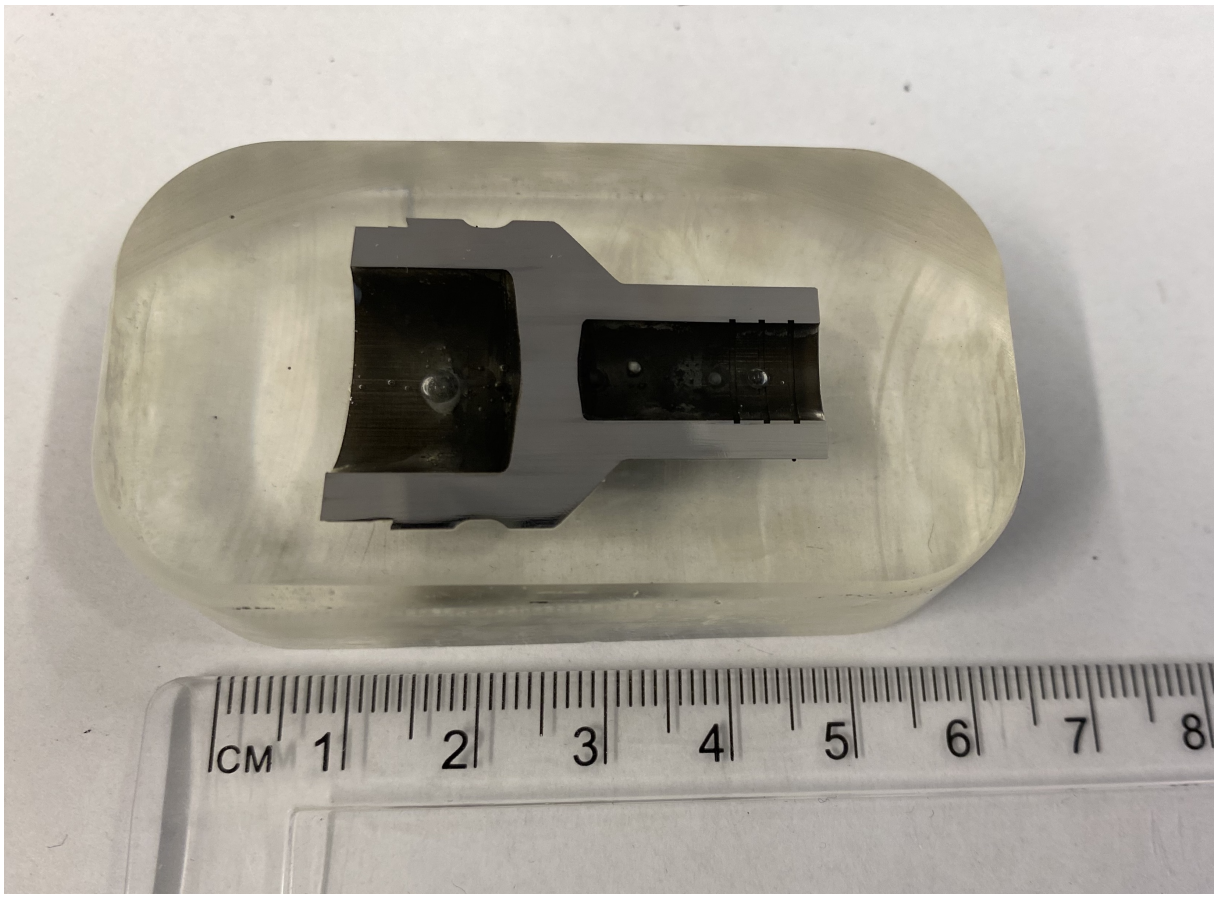


Figure 3.2: The steel sample in epoxy during EBSD preparation.

Subsequently, the sample was vibration polished for 2 hours with a 50nm silica suspension MasterMet 2, carried out in a Buehler Vibromet 2. The result is depicted in Figure 3.2.

The epoxy was removed by first sawing into the plastic and then gently removing smaller pieces. The remainder was left in hot acetone to swell and become brittle for around 4 hours. The rest was then broken up using a pipe wrench. All the epoxy was removed without the use of liquid nitrogen. Finally, it was sonicated in ethanol for 10 minutes, followed by a wash in soap and water, and lastly dried with an ethanol wash and a hairdryer. Plasma cleaning was performed on the specimen prior to the EBSD scan using a Fischione plasma cleaner.

3.3 EBSD Recording

EBSD recording was carried out utilizing a Zeizz Ultra 55 Field Emission SEM equipped with a NORDIF UF-1100 EBSD detector. The software version used at the time of recording was NORDIF 3.2.25.1. The parameters for the EBSD recording can be found in table 3.2 and table 3.3. The magnification level and scanning area were strategically selected to cover a martensite region bordered by ferrite bands on both sides. Five calibration patterns for pattern center optimization were recorded from ferrite regions on both sides of the EBSD area.

Table 3.2: EBSD settings.

Parameter	Value
Magnification	800
Acceleration voltage	20 kV
Working distance	23.4 mm
Tilt angle	70°
Step size	0.15 µs
Number of samples	485x106
Scan area	72.75×15.90 µm^2

Table 3.3: Acquisition and calibration settings.

Parameter	Acquisition	Calibration
Averaging	3	3
Frame rate	65 fps	150 fps
Resolution	240x240 px	160x160 px
Exposure time	15334 µs	6616 µs
Gain	5	5

3.4 Generating Master Patterns

The generation of a master pattern using EMsoft [17] involves a three-step process. Initially, one needs to create a crystal system file, also known as an `.xtal` file. This is followed by the Monte Carlo simulation of backscattered electrons, and concludes with the master pattern simulation. The process is described in detail by Østerhus, V. in [28].

Creating an `.xtal` file is achieved via EMsoft's command line `EMmkxtal`, with the necessary parameters provided. The specific parameters utilized for ferrite and martensite master pattern generation are listed in table 3.4 and table 3.5. Ferrite adopts a BCC crystal structure, giving it space group 229, while martensite has a BCT crystal structure, which can be described by space group 139. Both crystal systems make use of a Debye-Waller factor of 0.005 nm^2 [29].

The Monte Carlo simulations are conducted using the `EMMCOpenCL` command line, where the beam energies considered range from 10 to 20 keV. Following this, the master pattern simulation is executed with the `EMEBSDmaster` command. The result of this simulation is an `HDF5` file which contains square Lambert projections and stereographic projections of the master pattern.

Table 3.4: Parameters for creating `.xtal` file with EMsoft's `EMmkxtal`.

Phase	Ferrite	Martensite
Crystal system	1 (Cubic)	2 (Tetragonal)
Lattice parameters (nm)	a = 0.28610	Table 3.5
Space group	229; Im-3m	139; I4/mmm
Atomic number	26 (Fe)	26 (Fe)
Asymmetric atom position	(0, 0, 0)	(0, 0, 0)
Site occupation parameter	1.0	1.0
Debye-Waller factor (nm^2)	0.005	0.005

Table 3.5: Values for c and a lattice parameters in martensite at different concentrations of carbon. Calculated using Equation 2.4.

x (wt% C)	a (nm)	c (nm)	c/a
0.1	0.28597	0.28726	1.00451
0.2	0.28584	0.28842	1.00903
0.3	0.28571	0.28958	1.01355
0.4	0.28558	0.29074	1.01807
0.5	0.28545	0.29190	1.02260
0.6	0.28532	0.29306	1.02713
0.7	0.28519	0.29422	1.03166
0.8	0.28506	0.29538	1.03620
0.9	0.28493	0.29654	1.04075
1.0	0.28480	0.29770	1.04529
1.5	0.28415	0.30350	1.06810

3.5 Dictionary Indexing

The indexing was performed using dictionary indexing in EBSP Indexer powered by kikuchipy [4, 5]. Both static and dynamic backgrounds were eliminated from the dataset patterns. The data was not subjected to any offline averaging process. Table 3.6 lists the constant parameters from indexing runs. By varying pattern center, dictionary indexing provides vastly different results. The methods for the calibration of the pattern center are described in subsection 3.5.1.

Table 3.6: Indexing paramters that remained constant though several iterations of dictionary indexing.

Parameter	Value
Lazy Loading	False
Circular mask	True
Binning	1 (240x240px)
Angular step size	1.6°
Pattern center convention	TSL

As a part of Dictionary indexing, orientation refinement was performed. The refinement process aims to enhance the similarity between experimental patterns and simulated patterns that are projected from a master pattern by exploring the orientation space surrounding the the best normalized cross correlation (NCC) match from DI. Due to the fact that martensite has a tetragonal crystal structure and the slicing of the master patterns are discrete, orientation refinement should be done in three pseudo symmetrical orientations. Pseudo symmetry in the case of martensite/BCT means that the tetragonality is so small that the [100] and [010] zone axes can be mistaken for the [001] zone axis in an EBSD pattern and vice versa. In kikuchipy, refinement is carried out using the `refine_orientations` function on an indexed dataset. In the latest rounds of indexing (in combination with Method 3 and Method 4 for PC optimization, see section 3.5.1), the pseudo-symmetry operator is activated.

3.5.1 Pattern Center Calibration

Calibration of the pattern center required multiple iterations due to the insufficient results yielded by the first methods. The final method is supplemented to be able compare the open-source program EBSP Indexer to the commercial software EDAX/TSL

OIM Analysis.

Method 1: The most accessible method was deploying `pcopt.optimize` on calibration patterns, which was already incorporated into the EBSP Indexer. This tool for pattern center optimization is depicted in Figure 3.3. This method, like the other methods based on kikuchipy and pyEBSDindex, require an initial guess to run. The initial guess for the pattern center was EBSP Indexer's standard of (0.500, 0.800, 0.500), using the PC convention from TSL. The tuning function was repeatedly used until pattern center for each calibration pattern converged. The mean pattern center of the calibrated PCs obtained from the five calibration patterns, was used for indexing.

Method 2: The second alternative was to use ferrite and martensite patterns from the dataset, providing a larger selection of patterns with higher pixel resolution to be used for pattern center calibration. The patterns were selected from a 6x6 grid on the left side of the scan area, as shown in Figure 3.4. The methods used were `hough_indexing_optimize_pc` followed by `refine_orientation_projection_center`, both provided by kikuchipy. The initial estimate for the pattern center coordinates of (0.556, 0.780, 0.584) were based on the result from previous method. However, there were several shortcomings with this method, which are discussed in chapter 5.1.3.

Method 3: The third strategy for selecting calibration patterns involved picking individual high-quality ferritic patterns from both ends of the scan area. The chosen patterns are depicted in Figure 3.5. The initial estimate for pattern center coordinates was (0.527, 0.827, 0.572), a result from the previous method. Similar to the previous approach, `hough_indexing_optimize_pc` was used, followed by the function `refine_orientation_projection_center`. In addition to this, an outlier check was performed, excluding any outliers from the computation of the average pattern center. Outliers were identified as Hough indexed patterns with a normalized cross correlation metric match of less than 0.4.

Method 4: Lastly, the pattern center calibration method from EDAX/TSL OIM Data Collection (DC) on the calibration patterns was used. OIM DC doesn't necessitate an initial guess to perform pattern center optimization. The methodologies used in this process unfortunately remain undisclosed for the user.

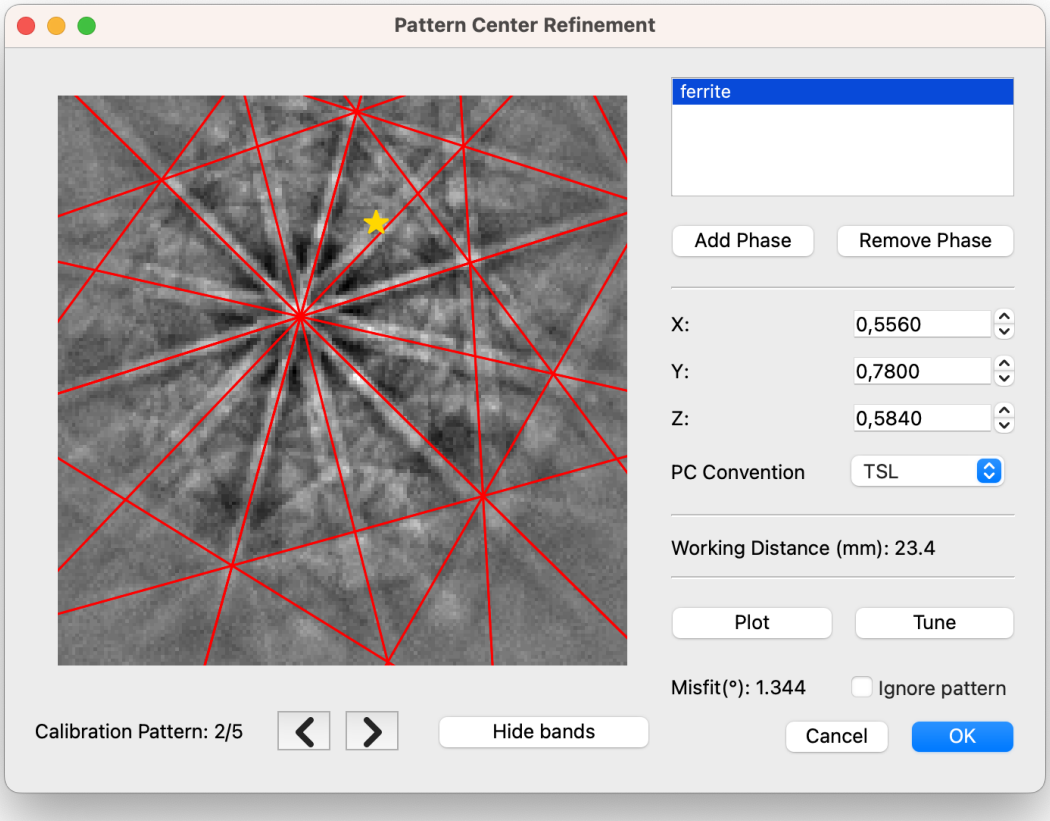


Figure 3.3: Pattern center optimization tool in EBSP Indexer, using `pcopt.optimize` from PyEBSDindex on individual patterns. Red lines indicate geometrical simulations with the provided pattern center.

3.6 Electron Probe Microanalysis

The steel sample underwent an electron probe microanalysis (EPMA) [30], employing the technique of wavelength dispersive X-ray spectroscopy (WDS) for precise elemental characterization. It was operated in spot mode, with a scanning time of 100 minutes per scan. The corresponding parameters for this procedure are summarized in Table 3.7. A series of three line scans were carried out, with the objective of measuring the concentrations of silicon, chromium, manganese, and carbon.

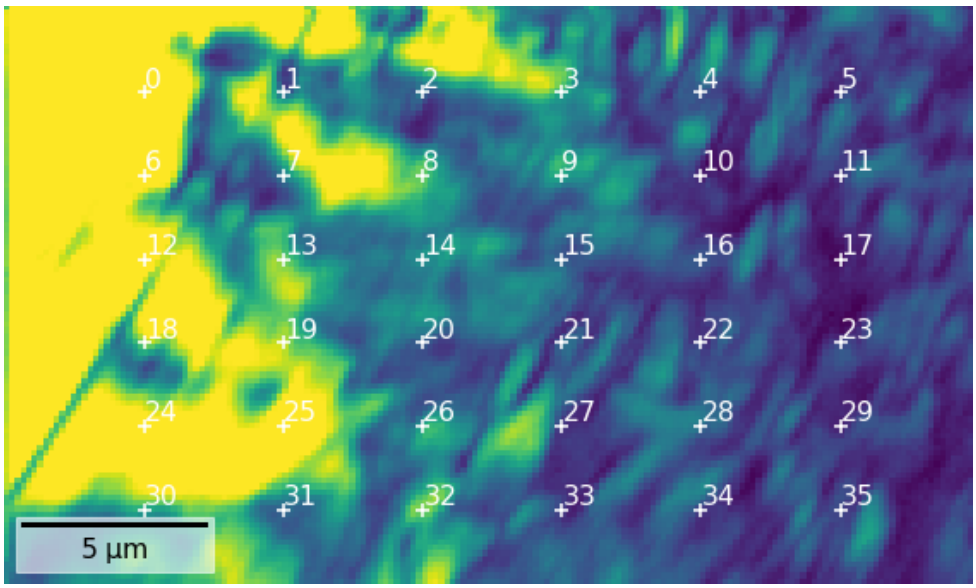


Figure 3.4: Grid of patterns used for pattern center optimization, Method 2.

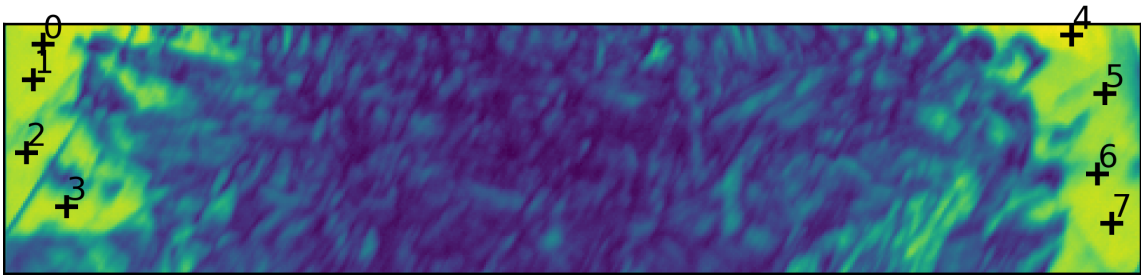


Figure 3.5: Patterns chosen for pattern center optimization, Method 3.

Table 3.7: Settings used for electron probe microanalysis.

Parameter	Value
Acceleration voltage	10 kV
Probe current	$3.0 \cdot 10^{-8}$ A
Dweel time	10 s
No. of points	601
Interval	1.00 µm
Line length	600.00 µm

Chapter 4

Results

This chapter presents the results derived from EBSD and EPMA analyses conducted on the mild steel sample. By tuning the parameters associated with the pattern center and the refinement process for dictionary indexing, a range of outcomes are displayed. Furthermore, the implementation of the EBSP Indexer software tool for pattern center optimization by use of pattern selection is covered in this chapter.

4.1 EBSD Indexing

After preparation for EBSD, the specimen underwent EBSD analysis. Figure 4.1 shows an orientation contrast image of the center of the specimen and the outlines the scanned area. Calibration pattern acquisition sites are clearly marked with yellow crosses.

Master patterns were generated using EMsoft [17], with tetragonality varying based on carbon concentrations ranging from 0.1 wt% to 1.5 wt%. Figure 4.2 showcases the master patterns of ferrite and martensite based on a 0.5 wt%C. The images picture on the fundamental sector of martensite and highlight the differences between ferrite and martensite.

By performing various pattern center optimization techniques, varied outcomes were observed when executing dictionary indexing. The techniques are described in detail in section 3.5.1. The corresponding PC coordinates are presented in Table 4.1.

The image quality map (IQ) of the EBSD area is presented in Figure 4.3. The white areas indicate high-quality patterns, typically associated with the ferrite phase [31]. Conversely, the darker regions indicate lower quality EBSPs, associated with martensite. Background correction, subtracting static and dynamic background, was applied before the generating of the image quality map.

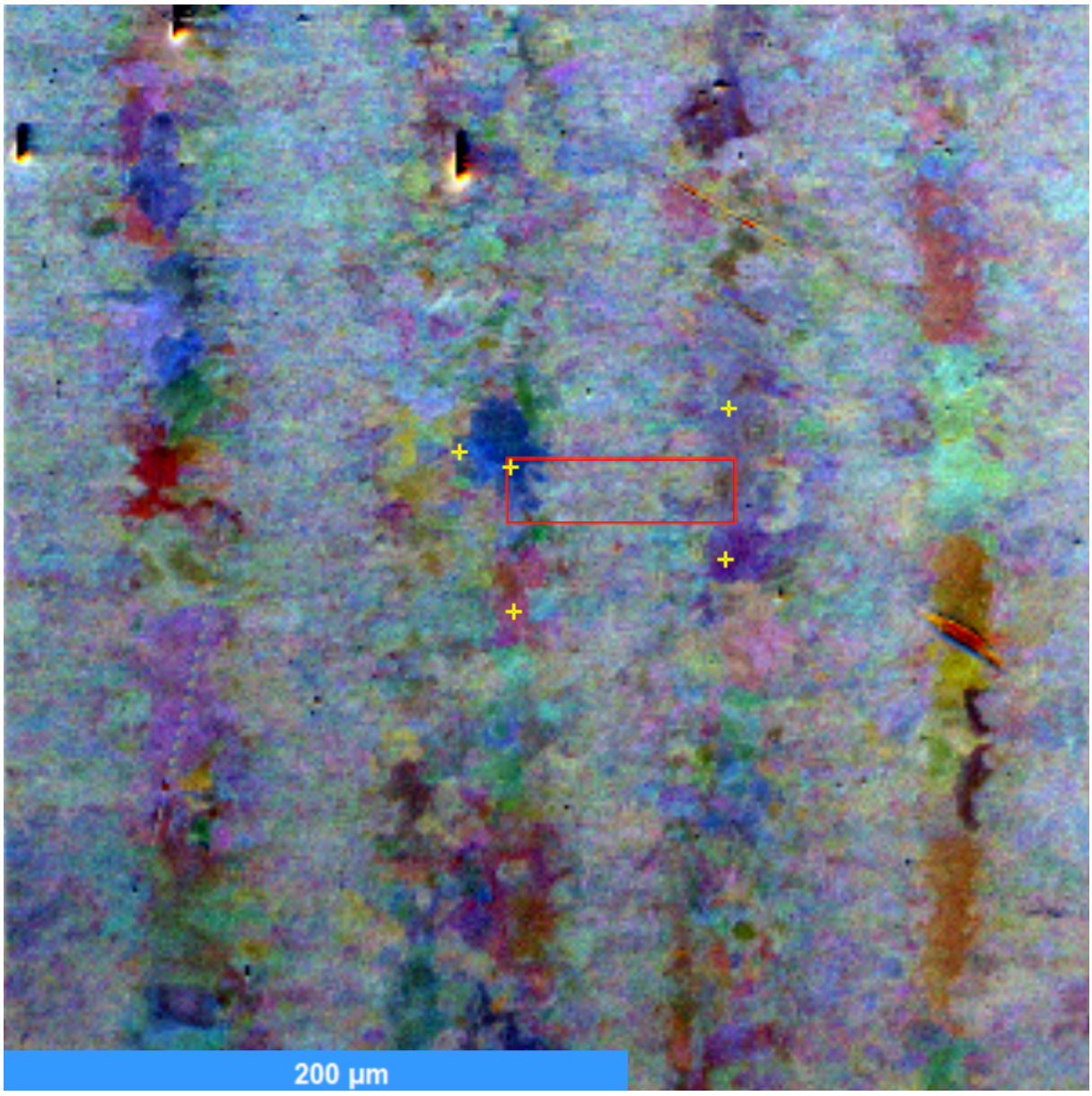


Figure 4.1: Color electron channelling contrast image of the specimen's central area. The region analyzed by EBSD is indicated with a red rectangle, and the positions of the calibration patterns are marked with yellow crosses.

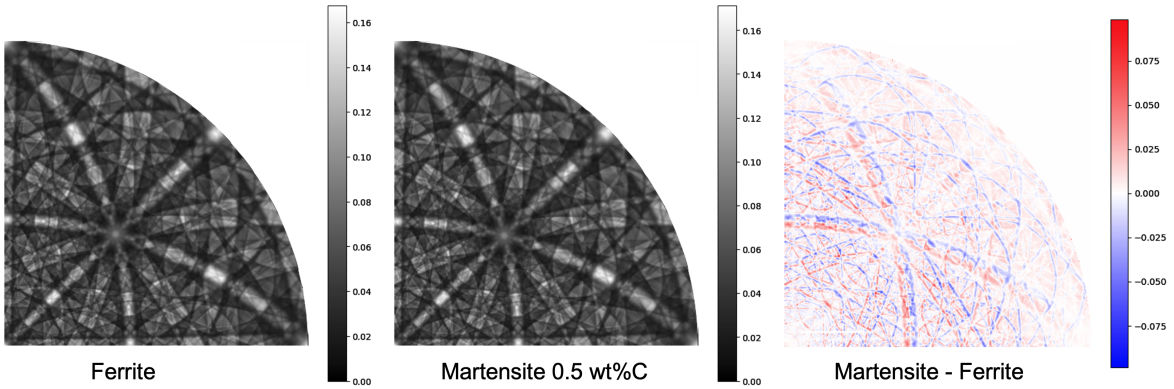


Figure 4.2: The figure showcases the master patterns representing ferrite and martensite with a carbon content of 0.5wt%, as well as the difference between the two. The red hue represents more intense martensite scattering, whereas the blue color signifies stronger ferrite scattering. The segment is restricted to the fundamental sector of martensite. The image is generated using kikuchipy [4].

Table 4.1: The pattern center coordinates (x*, y*, z*) utilized during indexing, derived through various methods. In brief terms, Method 1 refers to PC optimization by use of calibration patterns in EBSP Indexer, Method 2 uses a grid of martensite and ferrite patterns, Method 3 uses selected ferrite patterns from both sides of the EBSD area, and Method 4 obtains a PC from TSL Data Collection.

Refinement method	x*	y*	z*
Method 1	0.556	0.780	0.584
Method 2	0.527	0.827	0.572
Method 3	0.509	0.845	0.559
Method 4	0.518	0.860	0.566

Figure 4.3: Image quality map of EBSD area.

Despite the distinct phase maps displayed, dictionary indexing consistently produces the same result for the inverse pole figure (IPF) map across various pattern center and refinement techniques. The IPF map is depicted in Figure 4.4. To better present

the grain size and shape, the orientation color key connected with ferrite was utilized. The sub-ROI used for test-indexing is illustrated in Figure 4.5.

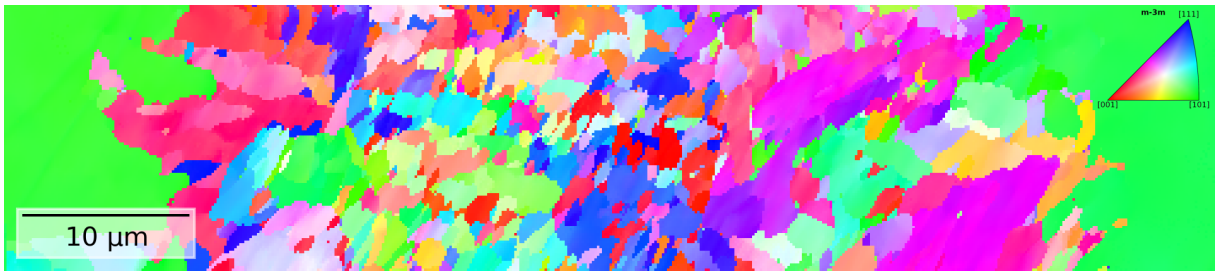


Figure 4.4: Inverse pole figure map of EBSD area.

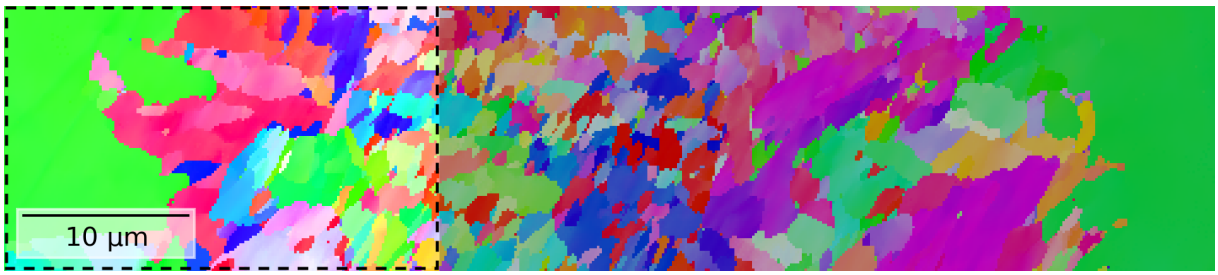
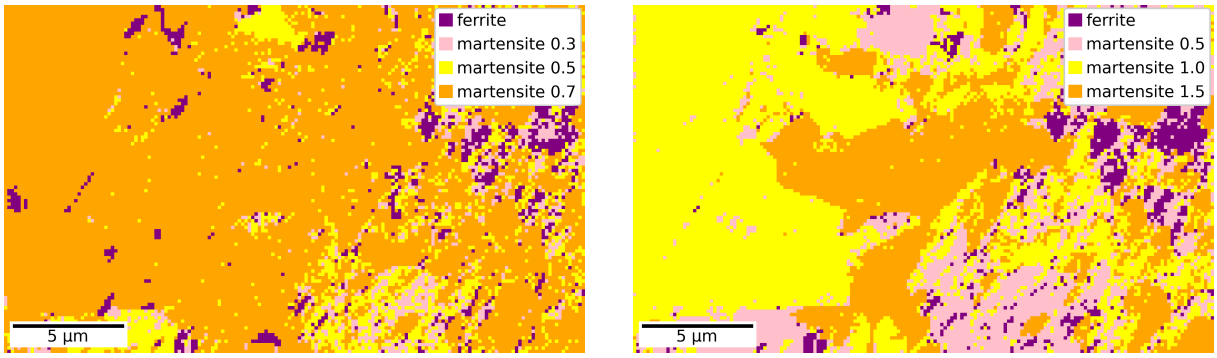


Figure 4.5: Inverse pole figure map, illustrating the area designated for test-indexing on the left-hand side of the domain.

Figure 4.6 presents two phase maps of the test area. Subfigures (a) and (b) deploy master patterns of ferrite and martensite with concentrations of up to 0.7 wt%C and 1.5 wt%C, respectively. A subordinate region of interest (sub-ROI) was utilized for test-indexing in this instance to improve the indexing speeds.



(a) **(b)**

Figure 4.6: Phase maps of the sub-ROI (Figure 4.5), using the pattern center derived from Method 1. Martensite phases are annotated based on their respective theoretical carbon concentrations, expressed in weight percentage (wt%).

By applying the second method of pattern center optimization, the phase map depicted in Figure 4.7 was attained. Ferrite and martensite master patterns with theoretical carbon concentrations of 0.3, 0.5, and 0.7 wt% were once again utilized for dictionary indexing. Figure 4.8 is obtained from DI using the pattern center derived from Method 3. The same master patterns as those in Figure 4.7 have been used and pseudo symmetry in martensite has not been accounted for, enabling a fair comparison between Method 2 and Method 3 PC optimization.

The phase maps depicted in Figure 4.9 and 4.10 were created utilizing the third method for pattern center optimization. The orientation refinement technique acknowledges martensitic pseudo symmetry, thereby reducing the portion of ferrite recognized in the martensite region. The indexing process utilizes different master patterns, with the final one exhibiting a smaller difference in tetragonality compared to the first.

Using the method for pattern center optimization provided by EDAX/TSL OIM Data Collection, the phase map in Figure 4.11 is generated after dictionary indexing. Just like Figure 4.10, martensite carbon concentrations go up to 0.5 wt%, and pseudo symmetry is accounted for in the refinement process.

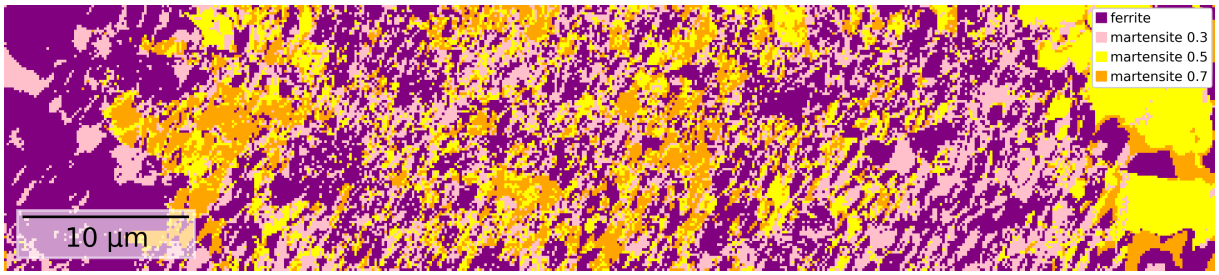


Figure 4.7: Phase map of the EBSD region, pattern center derived from Method 2. Martensite phases are denoted based on theoretical carbon concentrations (wt%).

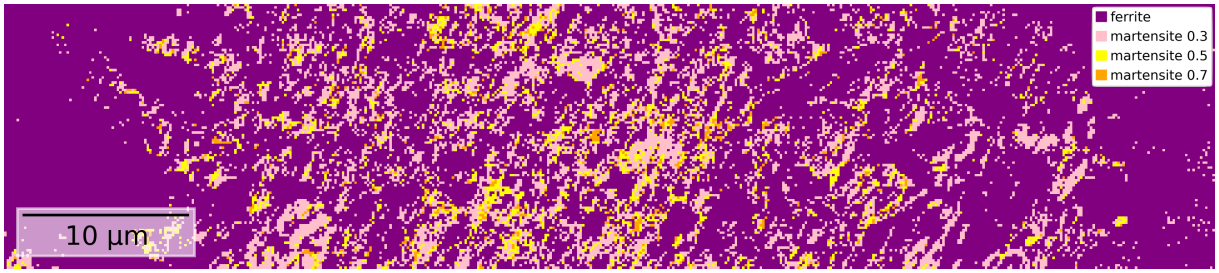


Figure 4.8: Phase map of the EBSD region, pattern center derived from Method 3. Pseudo symmetry operations were not utilized in the refinement process. Martensite phases are denoted based on their carbon concentrations (wt%).

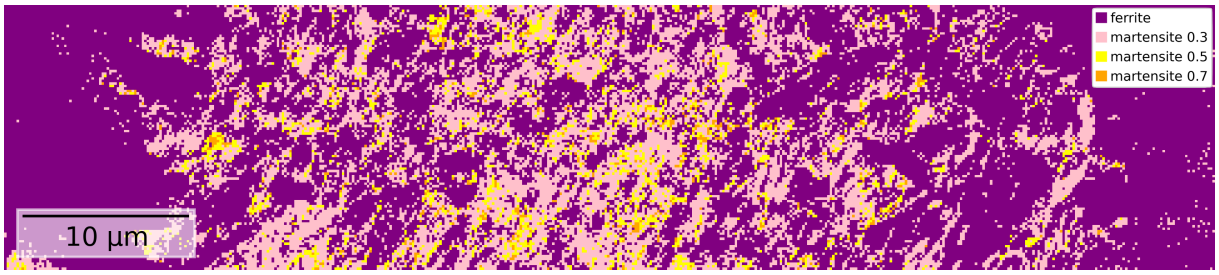


Figure 4.9: Phase map of the EBSD region, pattern center derived from Method 3. Pseudo symmetry operations were utilized in the refinement process. Martensite phases are denoted based on their carbon concentrations (wt%).

By using the PC optimization procedure from EDAX/TSL OIM Data Collection, the phase map portrayed in Figure 4.11 was produced. Precisely like in the previous method, this process utilized master patterns based on martensite with carbon concentrations up to 0.5 wt%. The orientation refinement accounted for pseudo symmetry.

The average zero-mean normalized cross correlation (NCC) for each method is shown in Table 4.2. The table incorporates all combinations of pattern centers, different master patterns, and refinement techniques, making them directly comparable to the Figures 4.6 - 4.11.

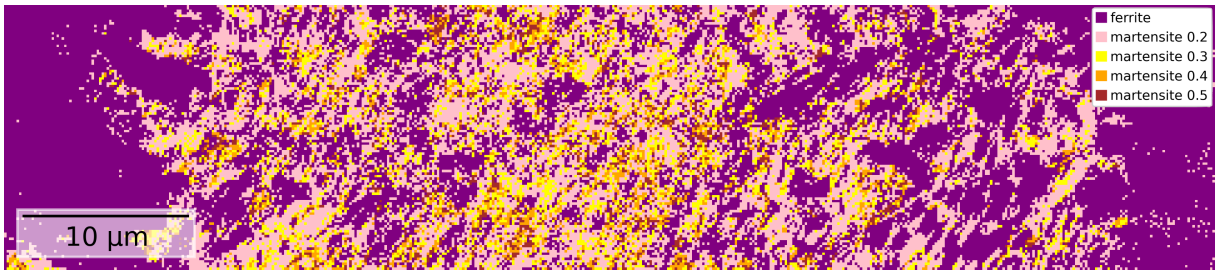


Figure 4.10: Phase map of the EBSD region, using the pattern center derived from Method 3. Pseudo symmetry operations were utilized in the refinement process. Martensite phases are denoted based on their carbon concentrations (wt%).

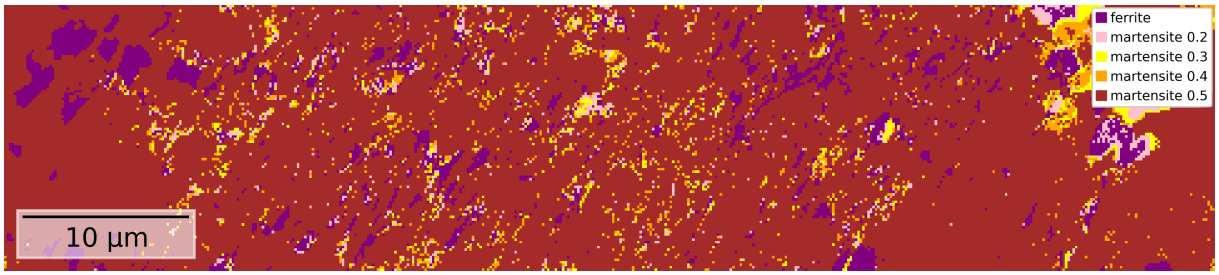


Figure 4.11: Phase map of the EBSD region, using the pattern center derived from Method 4. Pseudo symmetry operations were utilized in the refinement process. Martensite phases are denoted based on their carbon concentrations (wt%).

Table 4.2: The average NCC value for indexing corresponding to Figures 4.6 - 4.11. The rightmost column specifies which master patterns that were used and whether pseudo symmetry operations (PS ops) have been applied.

PC Method	Figure	Mean NCC	Comment
Method 1	4.6a	0.29351	MP wt%C $\leq$ 0.7
Method 1	4.6b	0.29711	MP wt%C $\leq$ 1.5
Method 2	4.7	0.38372	MP wt%C $\leq$ 0.7
Method 3	4.8	0.36849	MP wt%C $\leq$ 0.7
Method 3	4.9	0.36873	MP wt%C $\leq$ 0.7, PS ops
Method 3	4.10	0.36894	MP wt%C $\leq$ 0.5, PS ops
Method 4 (TSL)	4.11	0.34377	MP wt%C $\leq$ 0.5, PS ops

4.2 Electron Probe Microanalysis

Electron probe microanalysis (EPMA) was performed at the specimen's center, approximately in the EBSD zone. The operation settings are presented in Table 3.7. Three line scans were conducted, which are illustrated in a backscattered electron image (BEI), Figure 4.12. The Si, Cr, C, and Mn concentration profiles, as demonstrated in Figure 4.13, were derived from Line 2 of the analysis. A more detailed representation of the carbon concentration profile is presented in Figure 4.15. Utilizing the average concentration of every column of pixels from the EBSD phase map 4.10, an estimated concentration profile was obtained for the EBSD region. EBSD and EPMA carbon concentration profiles are displayed in Figure 4.14.

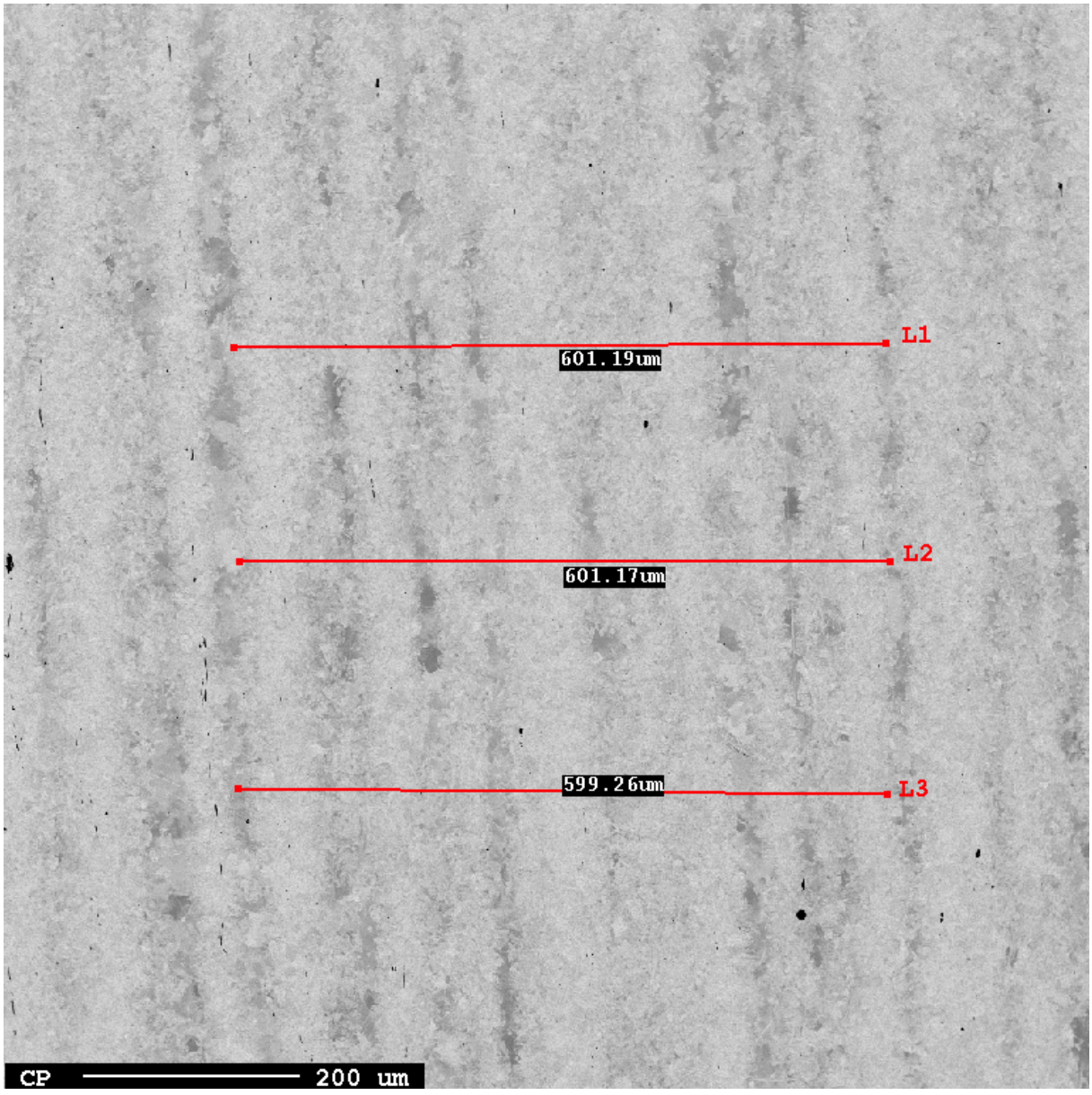


Figure 4.12: Backscattered electron image where the three lines used for the electron probe microanalysis are illustrated.

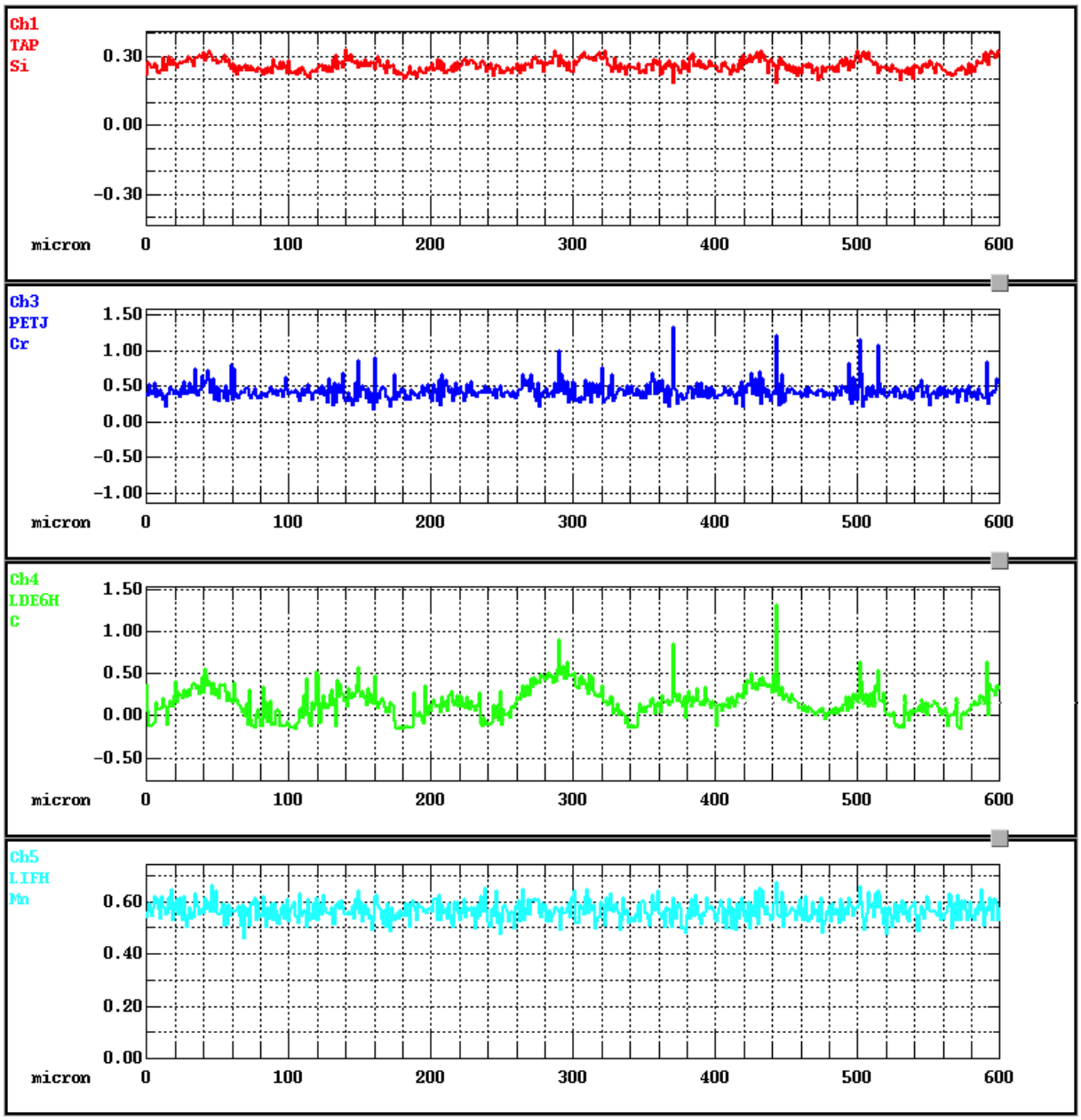


Figure 4.13: The information obtained from second line scan by electron probe micro-analysis. Each diagram corresponds to distributions of Si, Cr, C, and Mn concentrations in weight percentage across the analysed region.

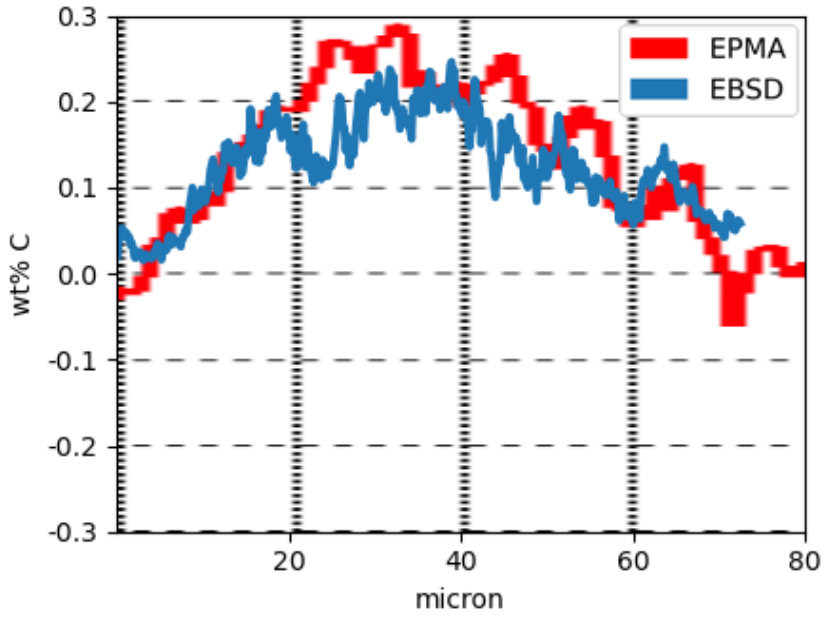


Figure 4.14: Carbon concentration profiles acquired from EPMA and EBSD. The concentration profile from EBSD was generated by calculating the average carbon concentration for each column of pixels in the phase map from Figure 4.10.

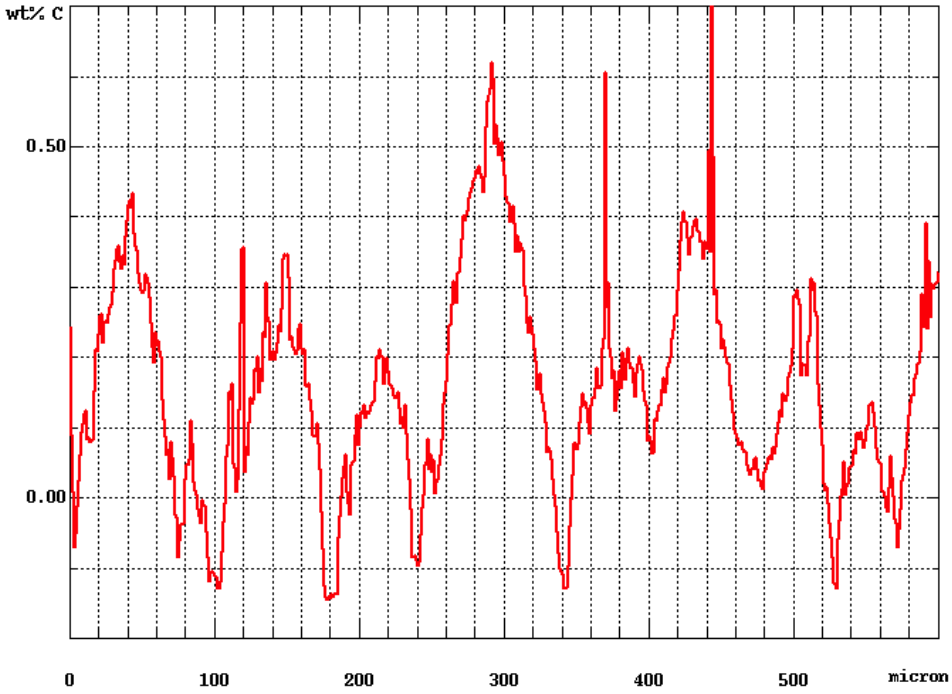


Figure 4.15: A detailed representation indicating the varying levels of carbon throughout the line scan, determined by EPMA.

4.3 EBSP Indexer – Pattern Center Optimization

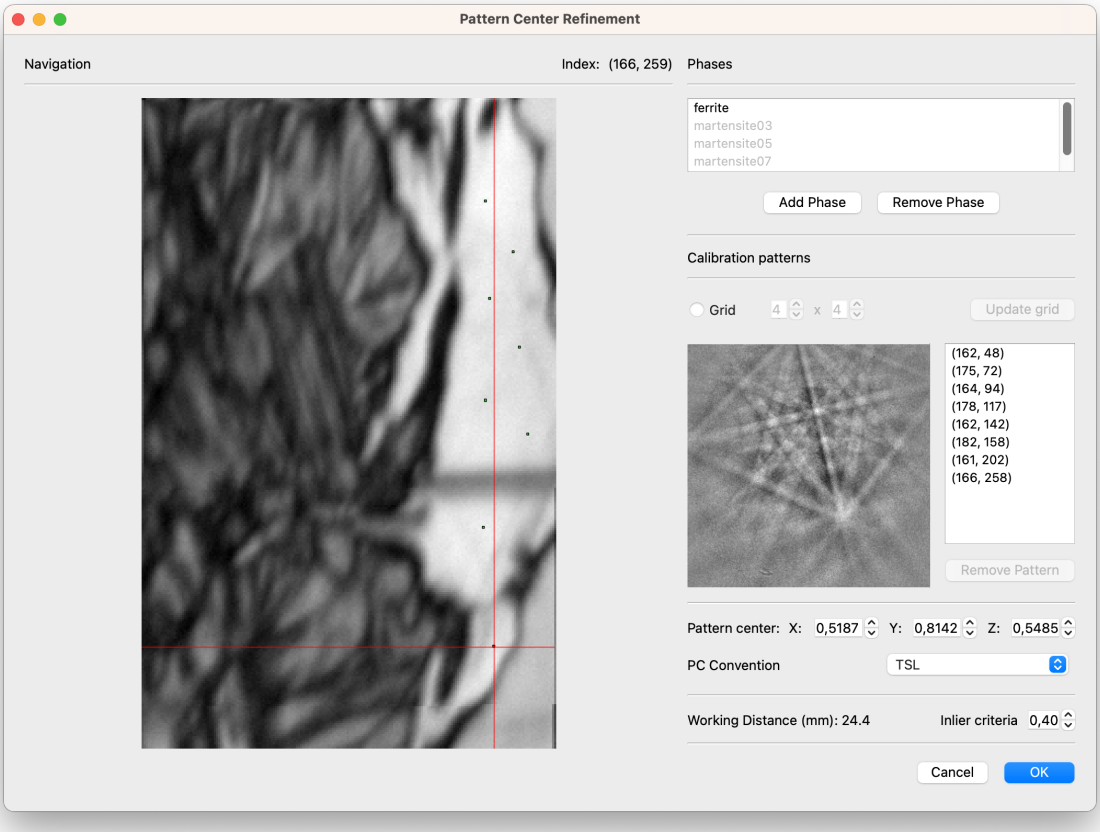


Figure 4.16: The pattern selection tool for PC optimization in EBSP Indexer.

The EBSP Indexer software was eventually updated to include the third method for pattern center optimization, which makes use of carefully selected EBSPs derived from the EBSD scan. The tool developed for optimizing pattern centers via pattern selection is displayed in Figure 4.16. The left section presents an image quality (IQ) map of the EBSD dataset. Users can navigate this IQ map with the mouse pointer, causing the corresponding EBSP to appear on the right. Calibration patterns can be conveniently selected with a simple click. These selected patterns are highlighted by a green square, and their respective indexes (x, y) are listed to the right of the EBSP display. Additionally, a grid option is available. It allows users to choose an array of calibration patterns, evenly spaced according to customizable dimensions.

The method of pattern center optimization in EBSP Indexer make use of master patterns to precisely determine phase parameters, and enables geometrical simulations. Master patterns can be incorporated using the "Add Phase" feature located at the top right of Figure 4.16, upon which they are registered in the list widget above.

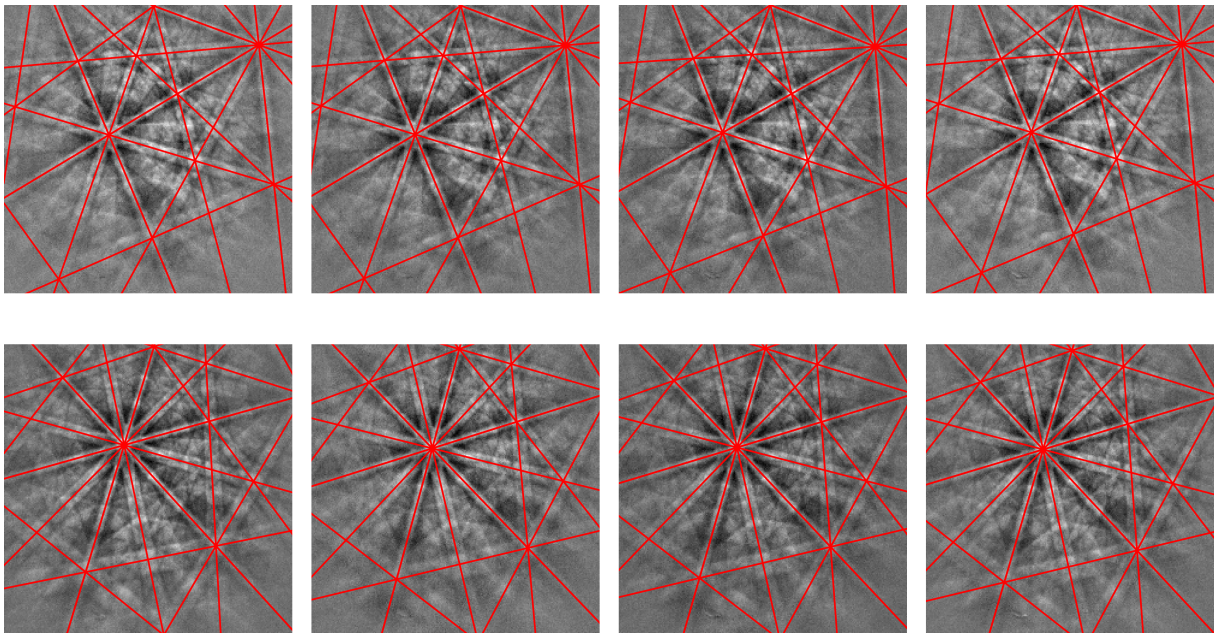


Figure 4.17: Geometrical simulations of the selected patterns. These patterns correspond to the ones chosen in Figure 3.5.

Pattern center calibration in EBSP Indexer requires an initial approximation of the pattern center to calculate the refined PC coordinates. The initial PC guess, in accordance to the desired PC convention, can be entered at the bottom right of the window. During the optimization process, a normalized cross correlation (NCC) match is assigned to each selected pattern. If the NCC value is lower than the *Inlier criteria*, the pattern and the associated pattern center are disregarded in the computation of the average. The average pattern center along with its standard deviation, the inlier, and outlier coordinates are stored in a `.txt` file. Visual representations of the chosen patterns, overlaid with geometrical simulations computed based on the optimized PC (as depicted in Figure 4.17), are preserved in a file. An IQ map that outlines the coordinates is also saved, example presented in Figure 3.5. Furthermore, a scatter plot depicting all the chosen patterns and their corresponding pattern centers after PC optimization is saved to a file, as illustrated by Figure 4.18.

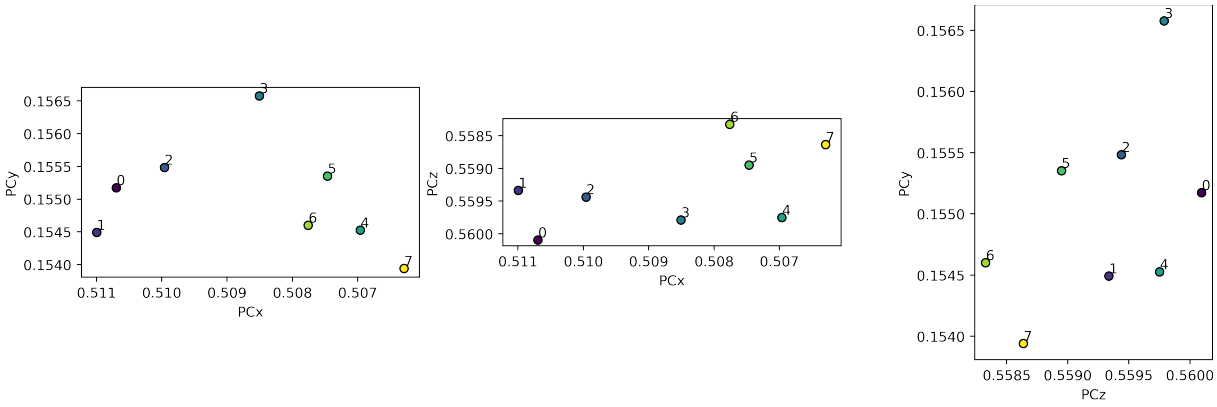


Figure 4.18: Scatter plot of the individual pattern centers obtained from pattern selection and optimization. By utilizing three figures, it becomes possible to visualize the coordinates of the pattern centers, (x*, y*, z*) and compare the coordinates with each other.

Chapter 5

Discussion

In this study, electron backscatter diffraction and dictionary indexing techniques were applied to investigate a mild steel specimen composed of ferrite and martensite. The objective was to acquire results that were in line with the observed microstructural characteristics, testing the limits of DI. The theoretical c/a relationship of martensite with 0.25 wt% carbon is 1.011, which is only a 1.1 % difference from ferrite. Master patterns for martensite was created with varying c/a ratios. The procedure was periodically evaluated, influencing future steps towards achieving the desired outcomes. Interestingly, the investigation revealed that pattern center calibration plays a critical role when performing dictionary indexing to distinguish between ferrite and martensite in mild steels. A range of pattern center optimization methods was experimented with through a process of iterative trial and error. The final results were compared to the carbon concentration profile obtained from electron probe microanalysis.

5.1 EBSD Analysis

The main factors influencing the outcomes of EBSD indexing revolve around master pattern simulation, pattern center calibration, and the refinement of orientations. In relation to the c/a ratio of martensite dependency on carbon concentrations, the existing literature presents contradictory findings, particularly when examining the effects at low concentrations – a circumstance that concern the specimen analyzed in this study. According to evidence from [32], no observable tetragonality was detected in martensite with a carbon content lower than 0.44 wt%, which exceeds the average carbon content in the mild steel specimen.

Given the evidence suggesting that martensite with low carbon concentrations exhibits negligible tetragonality, doubts were raised about the feasibility of procuring

meaningful data from dictionary indexing. Given that the master patterns crafted for this study do not explicitly employ the carbon content, but instead rely on the phase's lattice parameters, it remains justifiable to proceed with the investigation. It's worth noting that the carbon concentrations supporting these master patterns may exhibit some inconsistency compared to real world concentrations.

5.1.1 Master Patterns

The EMsoft open-source software was used to generate the master pattern used in this study [17]. The parameters necessary for this process rely on the crystal structure of martensite and ferrite. Much of this information, such as the crystal system and iron atom size, is well-documented in existing literature [18]. As for the lattice parameters, they were computed using findings from [3], which explores variations in martensite lattice parameters with a carbon concentration of 0.5 wt% and higher. The results demonstrated a distinct linear relationship between carbon content and tetragonality, as indicated by Figure 5.1.

It is important to understand that while each master pattern is predicated on the formula in Equation 2.4, what dictionary indexing (DI) essentially captures is the tetragonality present in martensite grains. Consequently, if a carbon content of less than 0.4 wt% results in no tetragonality, one could expect corresponding grains to be categorized as ferrite. As depicted in Figure 4.2, even the differences present in the master patterns of ferrite and martensite with carbon content as high as 0.5 wt% remain challenging to distinguish. The primary differentiation lies in the placement of the highest intensity, which seems to be slightly skewed towards the [001] direction for martensite.

5.1.2 Pre Processing

Before indexing, static background was removed from the raw, experimental patterns [4]. This static background was derived from an average of all patterns in the dataset. However, this method poses challenges when applied to scan areas that exhibit only a handful of crystallographic orientations. In such instances, the average of all patterns would still retain traces of Kikuchi lines, which, when removed from the patterns, could interfere with indexing. Given the extensive scan area, populated by a wide variety of grains, this issue will not affect the pre-processing, and the static background subtraction will function as intended.

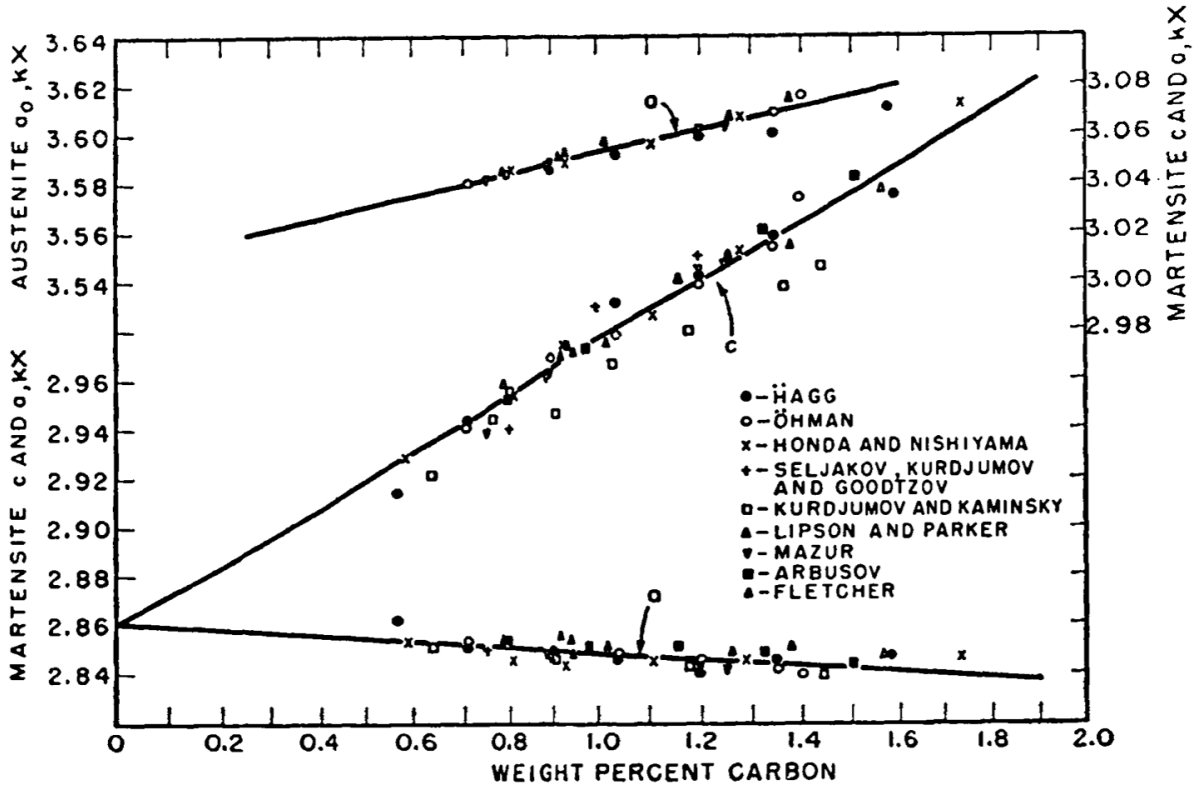


Figure 5.1: The relationship between lattice parameters of martensite and austenite relative to carbon content. The downmost line and the middle line shows the dependency of carbon concentration on the lattice parameters *a* and *c* for martensite, respectively. Figure from [3].

In contrast, dynamic background subtraction operates on individual patterns, eliminating a blurred version of the same pattern. Due to topographical differences, EBSPs possess different brightness, which can be eliminated by dynamic background correction. This method yields purely beneficial results, as it enhances contrast and ensures a more consistent intensity across the patterns.

A common pre-processing technique often used prior to dictionary indexing is the averaging of neighboring patterns to further enhance pattern quality. However, this approach encounters difficulties when applied to patterns near grain boundaries [20]. These boundary-adjacent patterns draw information from both grains, rendering them more difficult to index correctly. Given that the martensite phase is composed of numerous small grains, averaging of neighbouring patterns was not applied in order to prevent this potential issue.

5.1.3 Pattern Center Optimization

Four different methods were implemented for optimizing the pattern center. Method 1 was EBSP Indexer pattern center calibration with calibration patterns, which yielded convincing results in terms of the inverse pole figure map, as depicted in Figure 4.4. However, this technique demonstrated a bias towards detecting martensite with significantly higher carbon content than anticipated, presented in Figure 4.6. If the pattern center is even slightly misaligned, the gnomonic projection could distort the simulated patterns. Given the high resemblance among martensite phases, as illustrated in Figure 4.2, minor distortion in the reference pattern could result in a better normalized cross correlation (NCC) match for the incorrect phase. The noticeable grain boundary in Figure 4.6b indicates that dictionary indexing was capable of differentiating between BCT phases with different tetragonality, but the parameters were not accurately calibrated in line with experimental data.

The main limitations of Method 1 of PC optimization is its tendency to calibrate towards a local NCC maximum for each calibration pattern (not necessary the global), and that it depends heavily on the user's experience in comparing calibration patterns and geometric simulations. It's also worth noting that this method might be the user's first attempt at finding the pattern center, implying that the initial guess could be significantly off, thus allowing the software to choose a local optimization that greatly deviates from the actual pattern center.

Method 2 for pattern center calibration incorporates both ferrite and martensite patterns in a grid, as illustrated by Figure 3.4. PyEBSDindex's pattern center optimization make use of Hough indexing, tuning the pattern center around an initial PC prediction until the best match is located [23]. However, as the Hough indexing in pyEBSDindex exclusively supports cubic crystal structures (FCC and BCC), this method reveals its first weakness. Martensite patterns were not supported, and the results should not be trusted. Furthermore, Method 2 does not account for outliers, meaning that lower quality patterns contribute to the pattern center calculation.

As Table 4.2 illustrates, Method 2 yields the highest mean NCC value, likely due to the chosen patterns being of martensitic origin. Like kikuchipy's approach to dictionary indexing, this method got PC calibration utilizes Nelder-Mead optimization to identify the best match [4]. While the second method may identify a pattern center that output the highest average NCC value, it operates under the false assumption that martensite has the same BCC structure as ferrite, which is not the case. This indicates that the NCC metric shouldn't be relied upon exclusively. Indications of flawed indexing can be seen on the right side of Figure 4.7, which is indexed as a martensite grain with 0.5 wt% carbon, even though ferrite is the expected phase. The carbon content does not seem to correlate with the wave-like shape seen in the carbon concentration profiles obtained by EPMA, as shown in Figure 4.15.

Method 3 only uses ferritic patterns, which pyEBSDindex supports, and also excludes outliers. This method optimizes the alignment of the pattern center in respect to the ferrite phase. No outliers were discarded during the refinement of the pattern center, which was a consequence of selecting high-quality ferritic patterns and a sensible initial guess, as detailed in 3.5.1. Ferritic patterns were selected from both sides of the scan area, enhancing the optimization by considering the entire indexing area and featuring patterns from various grains with different orientations.

The final technique utilizes the PC refinement tool bundled with EDAX/TSL OIM Data Collection. Method 4 bases pattern center optimization on calibration patterns obtained from the SEM. However, the mechanisms driving this optimization are not communicated to the user. The results, demonstrated in Figure 4.11, show similar issues as Method 1, where it appears that distortions in projected simulated patterns causes a bias towards phases with higher tetragonality.

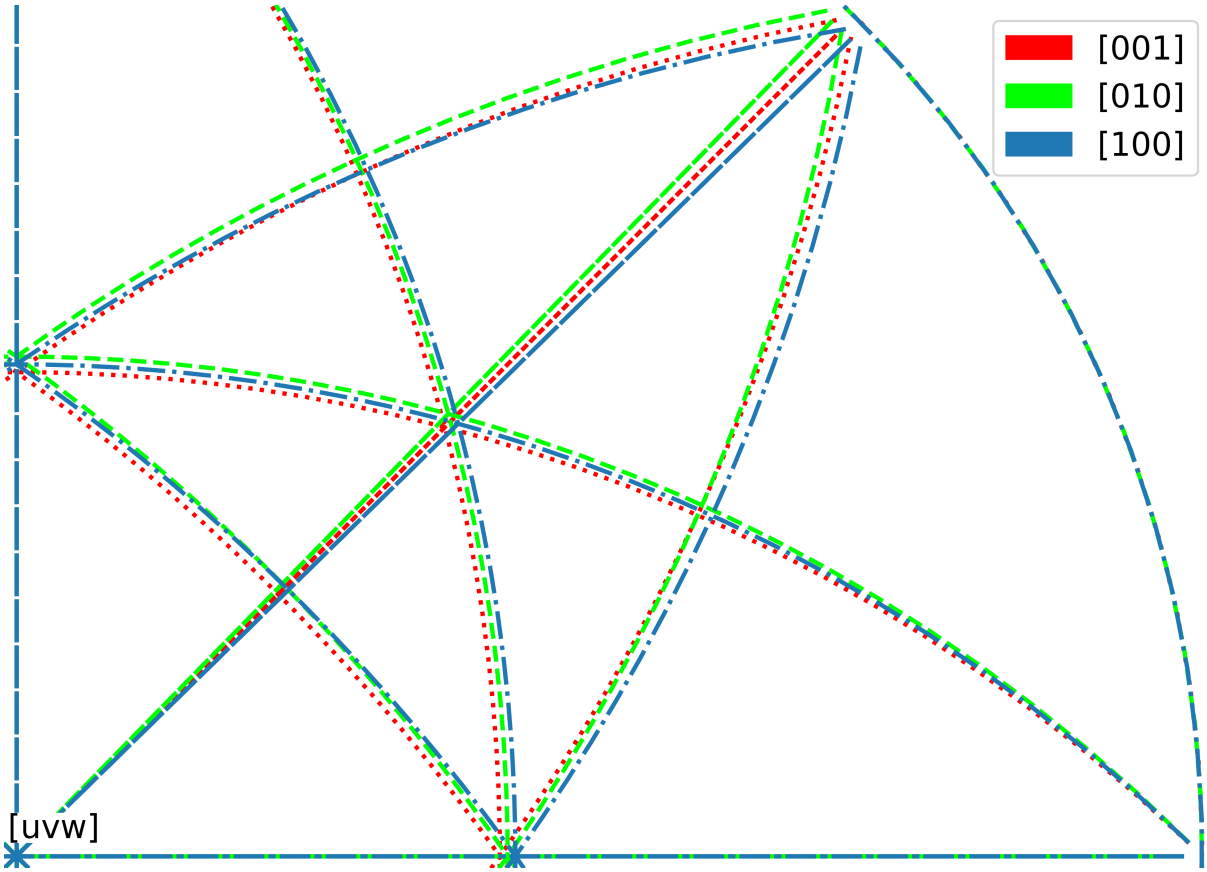


Figure 5.2: The zone axes from martensite with 0.5 wt%C, illustrating the pseudo-symmetry within the EBSPs. Each of the three pseudo-symmetrical orientations is represented by a distinct color. The label specifies the crystallographic direction [uvw] for the stereographic projection.

5.1.4 Refinement

In the case of martensite, backscattered electrons from three crystallographic directions display similar diffraction patterns. This zone axis difference is illustrated in Figure 5.2. The main consequence of this is that patterns initially identified as ferrite might have a better fit with martensite after refinement with pseudo-symmetrical operations. This is clear from examining Figures 4.8 and 4.9. An interesting observation is that this phenomenon does not reciprocate; using this method of refinement does not convert patterns indexed as martensite to ferrite. This could be explained by the fact that the NCC value for ferrite remains constant, and the highest NCC value for martensite cannot decrease after refinement, it may only increase.

5.2 Electron Probe Microanalysis

EPMA was utilized to gain deeper insights into the chemical mechanisms within the region of interest [30]. The phase map generated using the second pattern center refinement method, Figure 4.7, indicates high carbon concentrations near the ferrite grains boundary. To further investigate this, EPMA was deployed. Rather than an increase near the ferrite border, the carbon profile displays a continuous wave-like shape, indicating potential inaccuracies in the EBSD indexing. The concentrations of other elements do not exhibit such periodic tendencies, but align with the manufacturer's specifications (Table 3.1). The Mn concentration is slightly lower than specified, but it is not critical for this research.

Contrary to the results from Method 2 for PC optimization, Method 3 produced results that closely aligned with expectations and could now be compared to the concentration profile acquired from EPMA. By taking the average concentration of each column of pixels in phase map 4.10, a concentration profile was estimated for the EBSD region. Figure 4.14 shows that both the amplitude and frequency match the wave-like tendencies of a selected interval on the EPMA carbon profile.

5.3 EBSP Indexer – Pattern Center Optimization

Optimizing the pattern center through pattern selection was a crucial component in enhancing the project's results and hence, was incorporated into the EBSD Indexer software [5]. Dictionary indexing offers superior advantages over Hough indexing in three critical respects: it can effectively manage low-resolution EBSPs, index patterns close to or on grain boundaries, and distinguish between phases exhibiting similar microstructures, like aluminum and silicon, or ferrite and martensite. Obtaining an accurate pattern center significantly improves the efficacy of phase detection.

The pattern center optimization tool employs a familiar graphical design that aligns with the rest of the program. The navigation of the image quality map, which allows for real-time visualization of EBSPs, is based on the signal navigation tool detailed in [25]. To ensure the reliability of the optimization process, a dataset of high-quality patterns is necessary.

Chapter 6

Conclusion

This study aimed to explore the potential of EBSD dictionary indexing by distinguishing between martensite and ferrite in a mild steel sample. An few EBSD datasets were acquired, and EPMA concentration profiles were collected for the validation of the indexing results. As a part of this project, a software tool for accurate optimization of pattern center was implemented in EBSP Indexer, the open-source EBSD analysis software based on kikuchipy, developed for Mac and Windows. Based on the experiments conducted and the following analysis, the following conclusions can be drawn.

- The mild steel sample shows evidence of a ferritic and martensitic microstructure, based on the grain size, image quality map and carbon profile obtained by electron probe microanalysis.
- EBSD dictionary indexing can be used to distinguish martensite from ferrite in mild steels. Essential for great indexing results are high quality patterns and accurate indexing parameters, including the pattern center.
- Pattern center optimization can be done using different methods, from commercial and open-source software. The most successful results for phase determination were obtained from pattern center optimization in kikuchipy, using numerous high quality ferritic patterns. The PC optimization method was implemented in EBSP Indexer to make it more accessible, with a user-friendly interface. With this method in the software, scientists may easily obtain an accurate pattern center when working with similar challenges.
- Pseudo symmetry in martensite complicates the process of indexing martensite mild steels. The number of patterns indexed as martensite increases as a result of considering the pseudo symmetry of martensite.

- Despite literature suggesting that martensite with very low carbon concentrations does not exhibit significant tetragonality, the findings from this study indicate that a c/a relationship as low as c/a = 1.01 can be associated with a certain carbon concentration. Moreover, these results demonstrate a strong resemblance to the concentration profile obtained through EPMA analysis.

Chapter 7

Further Work

The methods adopted in this study sets the stage for future research. An outline highlighting promising areas of investigation is described below.

- The pattern center optimization method that gave the most reliable results, relies on high quality patterns from phases with cubic crystal structure. Using DI where there is absence of cubic phases, requires further research on precise pattern center refinement methods.
- An important factor when performing dictionary indexing on BCT structures is to take pseudo symmetrical orientations into account. kikuchipy has implemented this functionality, and it should be implemented in EBSP Indexer as well to be more versatile when dealing with similar indexing challenges.
- The raw data could still be improved to give dictionary indexing more information to work with. The EBSD computer could not handle the load of acquiring 320×320 pixels EBSPs. Choosing a higher resolution and increasing on-line averaging could diminish inaccuracy that originate from EBSD acquisition.
- Dictionary indexing with high resolution patterns and low angular step size is really slow, especcially when dealing with a large number of complex phases. kikuchipy does all this processing on the CPU, but GPUs are more suitable for image processing. Implementing DI that can run on the GPU is a step in the right direction for making DI even more accessible.

Bibliography

[1] Adam J. Schwartz. *Electron Backscatter Diffraction in Material Science*. 2nd ed. New York: Springer, 2009. Chap. 1.

[2] Patrick G. Callahan and Marc De Graef. "Dynamical Electron Backscatter Diffraction Patterns. Part I: Pattern Simulations". In: *Microscopy and Microanalysis* 19.5 (2013), pp. 1255–1265. DOI: https://doi.org/10.1017/S1431927613001840.

[3] C. S. Roberts. "Effect of Carbon on the Volume Fractions and Lattice Parameters Of Retained Austenite and Martensite". In: *Journal of Microscopy* 5.2 (Feb. 1953), pp. 203–204. ISSN: 1543-1851. DOI: https://doi.org/10.1007/BF03397477.

[4] Håkon Wiik Ånes et al. *pyxem/kikuchipy: kikuchipy 0.8.5*. Version v0.8.5. May 2023. DOI: 10.5281/zenodo.7954464. URL: https://doi.org/10.5281/zenodo.7954464.

[5] Erlend Mikkelsen Østvold, Hallvard Tangvik Tellefsen Relling, and Olav Leth-Olsen. *EBSP Indexer*. Version 0.1.0. May 2023. DOI: 10.5281/zenodo.7925262. URL: https://doi.org/10.5281/zenodo.7925262.

[6] Håkon Wiik Ånes. *figures*. 2020. URL: %5Curl%7Bhttps://github.com/hakonanes/figures%7D.

[7] Aimo Winkelmann et al. "Many-beam dynamical simulation of electron backscatter diffraction patterns". In: *Ultramicroscopy* 107.4 (2007), pp. 414–421. ISSN: 0304-3991. DOI: https://doi.org/10.1016/j.ultramic.2006.10.006. URL: https://www.sciencedirect.com/science/article/pii/S0304399106001975.

[8] J Goulden, P Trimby, and A Bewick. "The Benefits and Applications of a CMOS-based EBSD Detector". In: *Microscopy and Microanalysis* 24.S1 (Aug. 2018), pp. 1128–1129. ISSN: 1431-9276. DOI: 10.1017/S1431927618006128. eprint: https://academic.oup.com/mam/article-pdf/24/S1/1128/48301969/mam1128.pdf. URL: https://doi.org/10.1017/S1431927618006128.

[9] H W Ånes et al. "Processing and indexing of electron backscatter patterns using open-source software". In: *IOP Conference Series: Materials Science and Engineering* 891.1 (July 2020), p. 012002. DOI: 10.1088/1757-899X/891/1/012002. URL: https://dx.doi.org/10.1088/1757-899X/891/1/012002.

[10] J. Hjelen et al. "EBSP, Progress in Technique and Applications". In: *Texture, Stress and Microstructures* 20 (1993). ISSN: 1687-5397. DOI: 10.1155/TSM.20.29. URL: https://doi.org/10.1155/TSM.20.29.

[11] Edward L. Pang, Peter M. Larsen, and Christopher A. Schuh. "Global optimization for accurate determination of EBSD pattern centers". In: *Ultramicroscopy* 209 (2020), p. 112876. ISSN: 0304-3991. DOI: https://doi.org/10.1016/j.ultramic.2019.112876. URL: https://www.sciencedirect.com/science/article/pii/S030439911930292X.

[12] Håkon Wiik Ånes. *kikuchipy User Guide*. Last accessed 25 May 2023. 2023. URL: https://kikuchipy.org/en/stable/user/index.html.

[13] N. C. Krieger Lassen. "Automatic high-precision measurements of the location and width of Kikuchi bands in electron backscatter diffraction patterns". In: *Journal of Microscopy* 190.3 (1998), pp. 375–391. DOI: https://doi.org/10.1046/j.1365-2818.1998.00330.x. URL: https://onlinelibrary.wiley.com/doi/pdf/10.1046/j.1365-2818.1998.00330.x.

[14] Chaoyi Zhu, Kevin Kaufmann, and Kenneth Vecchio. "Automated Reconstruction of Spherical Kikuchi Maps". In: *Microscopy and Microanalysis* 25.4 (2019), pp. 912–923. DOI: 10.1017/S1431927619000710.

[15] Gonzalez Rafael C, Woods Richard E, et al. *Digital image processing*. Prentice Hall, 2002.

[16] Gert Nolze, Ralf Hielscher, and Aimo Winkelmann. "Electron backscatter diffraction beyond the mainstream". In: *Crystal Research and Technology* 52.1 (2017), p. 1600252.

[17] Marc De Graef. *EMsoft*. Version v5.0.0. Apr. 2023. URL: https://doi.org/10.5281/zenodo.3489720.

[18] W.B. Pearson, P. Villars, and L.D. Calvert. *Pearson's Handbook of Crystallographic Data for Intermetallic Phases*. Pearson's Handbook of Crystallographic Data for Intermetallic Phases. American Society for Metals, 1985.

[19] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2005. ISBN: 0387212396.

[20] Stuart I. Wright et al. "Introduction and comparison of new EBSD post-processing methodologies". In: *Ultramicroscopy* 159 (2015), pp. 81–94. ISSN: 0304-3991. DOI: https://doi.org/10.1016/j.ultramic.2015.08.001. URL: https://www.sciencedirect.com/science/article/pii/S0304399115300188.

[21] Francisco de la Peña et al. *hyperspy/hyperspy: Release v1.7.1*. Version v1.7.1. June 2022. DOI: 10.5281/zenodo.6659919. URL: https://doi.org/10.5281/zenodo.6659919.

[22] Håkon Wiik Ånes et al. *pyxem/orix: orix 0.8.0*. Version v0.8.0. Dec. 2021. DOI: 10.5281/zenodo.5795877. URL: https://doi.org/10.5281/zenodo.5795877.

[23] Dave Rowenhorst. *pyEBSDIndex documentation*. Last accessed 25 May 2023. 2023. URL: https://pyebsdindex.readthedocs.io/en/latest/.

[24] The Qt Company. *Qt for Python*. 2023. URL: https://doc.qt.io/qtforpython-6/.

[25] Hallvard Relling. "EBSP Indexer - using an open-source dictionary indexing software for phase differentiation". MA thesis. Norwegian University of Science and Technology, 2023.

[26] Erlend M. Østvold. "EBSP Indexer - An open-source alternative to commercial EBSD software". MA thesis. Norwegian University of Science and Technology, 2023.

[27] W.D. Callister and D.G. Rethwisch. *Materials Science and Engineering: An Introduction, 9th Edition: Ninth Edition*. John Wiley and Sons, Incorporated, 2013. ISBN: 9781118476543.

[28] Vetle Østerhus. "Improving Quantification of Sigma and Chi Phases in SDSS with EBSD". In: (June 2020), Appendix A: I–V.

[29] L.-M. Peng et al. "Debye–Waller Factors and Absorptive Scattering Factors of Elemental Crystals". In: *Acta Crystallographica Section A* 52.3 (May 1996), pp. 456–470. DOI: 10.1107/S010876739600089X. URL: https://journals.iucr.org/a/issues/1996/03/00/zh0008/zh0008_82472sup1.pdf.

[30] Peter Schiffman and Sarah Roeske. "Electron Microprobe Analysis of Minerals". In: *Encyclopedia of Physical Science and Technology (Third Edition)*. Ed. by Robert A. Meyers. Third Edition. New York: Academic Press, 2003, pp. 293–306. ISBN: 978-0-12-227410-7. DOI: https://doi.org/10.1016/B0-12-227410-5/00211-8. URL: https://www.sciencedirect.com/science/article/pii/B0122274105002118.

[31] Bong-Yong Jeong, Raynald Gauvin, and S. Yue. "EBSD Study of Martensite in a Dual Phase Steel". In: *Microscopy and Microanalysis* 8 (Aug. 2002), pp. 700–701. DOI: 10.1017/S1431927602106507.

[32] Tomohito Tanaka et al. "Tetragonality of Fe-C martensite – a pattern matching electron backscatter diffraction analysis compared to X-ray diffraction". In: *Acta Materialia* 195 (2020), pp. 728–738. ISSN: 1359-6454. DOI: https://doi.org/10.1016/j.actamat.2020.06.017. URL: https://www.sciencedirect.com/science/article/pii/S1359645420304419.

Appendix A

EBSP Indexer: Pattern Center Optimization - Pattern Selection

The following code demonstrates the mechanisms of the pattern selection tool used for pattern center optimization. This tool is implemented in the EBSP Indexer as a part of this master's thesis, where it is referred to as Method 3.

```
from math import ceil, floor
from os import devnull, mkdir, path

import dask as da
import kikuchipy as kp
import matplotlib.pyplot as plt
import numpy as np
from diffsims.crystallography import ReciprocalLatticeVector
from matplotlib.patches import Rectangle
from matplotlib.widgets import Cursor
from orix.crystal_map import PhaseList
from PySide6.QtCore import Qt
from PySide6.QtWidgets import QDialog, QDialogButtonBox

from scripts.pc_from_wd import pc_from_wd
from scripts.signal_loader import EBSDDataset
from ui.ui_pc_selection import Ui_PCSelection
from utils import FileBrowser, SettingFile, sendToJobManager

progressbar_bool = False

ALLOWED_SPACE_GROUPS = ["Fm-3m", "Im-3m"] # FCC, BCC

class PCSelectionDialog(QDialog):
    def __init__(self, parent=None, pattern_path=None):
        super().__init__(parent)
        # pattern path
        self.pattern_path = pattern_path

        # pattern name
```

```
31          self.pattern_name = path.basename(self.pattern_path)
32
33          # working directory
34          self.working_dir = path.dirname(self.pattern_path)
35
36          self.ui = Ui_PCSelection()
37          self.ui.setupUi(self)
38          self.setupConnections()
39          self.setupInitialSettings()
40
41          #self.fileBrowserOD = FileBrowser(FileBrowser.OpenDirectory)
42          self.fileBrowserOD = FileBrowser(
43              mode=FileBrowser.OpenFile,
44              filter_name="*.h5"
45          )
46          self.plot_navigator(0,0)
47          self.coordinates = []
48
49      def setupInitialSettings(self):
50          """
51          Reads parameters from project_settings.txt, advanced_settings.txt
52          """
53          self.setting_file = SettingFile(
54              path.join(self.working_dir, "project_settings.txt")
55          )
56          self.program_settings = SettingFile("advanced_settings.txt")
57
58          self.s = kp.load(self.pattern_path, lazy=True)
59          working_distance = self.s.metadata.Acquisition_instrument.SEM.working_distance
60          self.ui.workingDistanceLabel.setText("Working Distance (mm): "+str(working_distance))
61
62          try:
63              self.convention = self.setting_file.read("Convention")
64          except:
65              self.convention = self.program_settings.read("Convention")
66
67          self.ui.conventionBox.setCurrentText(self.convention)
68
69          try:
70              self.pc = eval(self.setting_file.read("PC"))
71
72          except:
73              try:
74                  microscope = self.s.metadata.Acquisition_instrument.SEM.microscope
75                  self.pc = pc_from_wd(microscope, working_distance, self.convention)
76              except:
77                  self.pc = (0.5000, 0.8000, 0.5000)
78
79          self.updatePCSpinBox()
80
81          self.mp_paths = {}
82
```

```
83          i = 1
84          while True:
85              try:
86                  mp_path = self.setting_file.read(f"Master pattern {i}")
87                  try:
88                      mp = kp.load(mp_path, lazy=True)
89                  except IOError as ioe:
90                      raise ioe
91                  if not len(mp.phase.name):
92                      mp.phase.name = path.dirname(mp_path).split("/").pop()
93                  self.mp_paths[mp.phase.name] = mp_path
94                  self.ui.listPhases.addItem(mp.phase.name)
95                  space_group = mp.phase.space_group.short_name
96                  if space_group not in ALLOWED_SPACE_GROUPS:
97                      self.ui.listPhases.item(i-1).setFlags(Qt.NoItemFlags)
98                  i += 1
99              except IOError as ioe:
100                 raise ioe
101             except:
102                 break
103
104
105         self.changeStateOfButtons()
106
107
108     def setupConnections(self):
109         self.ui.buttonAddPhase.clicked.connect(lambda: self.addPhase())
110         self.ui.buttonRemovePhase.clicked.connect(lambda: self.removePhase())
111         self.ui.buttonRemovePattern.clicked.connect(lambda: self.removePattern())
112         self.ui.listCoordinates.itemSelectionChanged.connect(lambda: self.coordinateListClicked())
113         self.ui.buttonGrid.clicked.connect(lambda: self.gridClicked())
114         self.ui.buttonUpdateGrid.clicked.connect(lambda: self.gridClicked())
115         self.ui.buttonBox.accepted.connect(lambda: self.runPatternCenterOpimization())
116         self.ui.buttonBox.rejected.connect(lambda: self.reject())
117         self.ui.conventionBox.currentTextChanged.connect(lambda: self.updatePCConvention())
118
119     # Phases
120     def addPhase(self):
121         if self.fileBrowserOD.getFile():
122             mp_path = self.fileBrowserOD.getPaths()[0]
123             try:
124                 mp = kp.load(mp_path, lazy=True)
125                 if not len(mp.phase.name):
126                     mp.phase.name = path.basename(path.dirname(mp_path))
127                 phase_name = mp.phase.name
128             except Exception as e:
129                 raise e
130             if phase_name not in self.mp_paths.keys():
131                 self.mp_paths[phase_name] = mp_path
132                 self.ui.listPhases.addItem(phase_name)
133
134             self.fileBrowserOD.setDefaultDir(path.dirname(mp_path))
```

```
135             space_group = mp.phase.space_group.short_name
136             if space_group not in ALLOWED_SPACE_GROUPS:
137                 self.ui.listPhases.item(len(self.mp_paths.keys())-1).setFlags(Qt.NoItemFlags)
138             else:
139                 self.phase = phase_name
140                 self.ui.listPhases.setCurrentRow(len(self.mp_paths.keys())-1)
141         self.changeStateOfButtons()
142 
143     def removePhase(self):
144         self.mp_paths.pop(str(self.ui.listPhases.currentItem().text()))
145         self.ui.listPhases.takeItem(self.ui.listPhases.currentRow())
146         self.ui.listPhases.clearSelection()
147 
148         self.changeStateOfButtons()
149 
150     # Patterns
151     def addPattern(self, x, y):
152         if [x, y] not in self.coordinates:
153             self.coordinates.append([x, y])
154             self.ui.listCoordinates.addItem(f"({x}, {y})")
155 
156         self.changeStateOfButtons()
157 
158     def removePattern(self):
159         index = self.ui.listCoordinates.currentRow()
160 
161         try:
162             self.click_rect.remove()
163         except:
164             pass
165 
166         self.ui.listCoordinates.takeItem(self.ui.listCoordinates.currentRow())
167         del self.coordinates[index]
168 
169         self.update_rectangles()
170         self.ui.MplWidget.canvas.draw()
171 
172         if len(self.coordinates) == 0:
173             self.ui.buttonRemovePattern.setEnabled(False)
174 
175         self.changeStateOfButtons()
176 
177     def coordinateListClicked(self):
178         try:
179             coords = self.ui.listCoordinates.currentItem().text().split(", ")
180             x, y = int(coords[0].strip("()")), int(coords[1].strip("()"))
181             self.plot_signal(x, y)
182         except:
183             x = -1
184 
185         try:
186             self.click_rect.remove()
```

```
187             except:
188                 pass
189 
190             if x >= 0:
191                 self.current_rectangle(x,y)
192 
193             self.ui.MplWidget.canvas.draw()
194 
195             self.ui.buttonRemovePattern.setEnabled(True)
196 
197         def gridClicked(self):
198             if self.ui.buttonGrid.isChecked():
199                 self.coordinates = []
200                 self.ui.listCoordinates.clear()
201                 grid_shape = [int(self.ui.spinBoxGridX.value()), int(self.ui.spinBoxGridY.value())]
202                 self.updateGridButtons(enable=True)
203                 s_grid, coords = self.s.extract_grid(grid_shape, return_indices=True)
204                 try:
205                     coords_converted = self.convertGridList(grid_shape, coords)
206                 except:
207                     print("Grid shape not supported for this dataset.")
208                 for coord in coords_converted:
209                     self.addPattern(coord[0], coord[1])
210 
211                 self.update_rectangles()
212                 self.ui.MplWidget.canvas.draw()
213             else:
214                 self.updateGridButtons(enable=False)
215 
216         def convertGridList(self, grid_shape, coords):
217             x, y = grid_shape[0], grid_shape[1]
218             new_list = [[0,0]]*(x*y)
219             for i in range(x):
220                 for j in range(y):
221                     new_list[j+i*y] = [coords[1][j][i]-1, coords[0][j][i]-1]
222             return new_list
223 
224         def updateGridButtons(self, enable):
225             self.ui.spinBoxGridX.setEnabled(enable)
226             self.ui.spinBoxGridY.setEnabled(enable)
227             self.ui.buttonUpdateGrid.setEnabled(enable)
228             self.ui.labelMultiplicator.setEnabled(enable)
229 
230         def changeStateOfButtons(self):
231             if bool(len(list(self.mp_paths.keys()))) and (self.ui.listCoordinates.count() != 0):
232                 enable = True
233             else:
234                 enable = False
235             self.ui.buttonRemovePhase.setEnabled(bool(len(list(self.mp_paths.keys()))))
236             self.ui.buttonBox.button(QDialogButtonBox.Ok).setEnabled(enable)
237 
238         def updatePCSpinBox(self):
```

```
239         self.ui.spinBoxX.setValue(self.pc[0])
240         self.ui.spinBoxY.setValue(self.pc[1])
241         self.ui.spinBoxZ.setValue(self.pc[2])
242 
243     def updatePCArrayFromSpinBox(self):
244         self.pc = (self.ui.spinBoxX.value(), self.ui.spinBoxY.value(), self.ui.spinBoxZ.value())
245 
246     def updatePCConvention(self):
247         self.convention = self.ui.conventionBox.currentText()
248 
249 
250 ### INTERACTION WITH NAVIGATOR ###
251 
252     def plot_navigator(self, x=0, y=0):
253         self.dataset = EBSDDataset(self.s)
254 
255         try:
256             self.ui.MplWidget.canvas.mpl_disconnect(self.cid)
257             self.ui.MplWidget.canvas.mpl_disconnect(self.hover_id)
258         except:
259             pass
260 
261         # plot to MplCanvas
262         self.ui.MplWidget.vbl.setContentsMargins(0, 0, 0, 0)
263         self.ui.MplWidget.canvas.ax.clear()
264         self.ui.MplWidget.canvas.ax.axis(False)
265 
266         self.cursor = Cursor(
267             self.ui.MplWidget.canvas.ax,
268             useblit=True,
269             color="red",
270             linewidth=1,
271             linestyle="-",
272             alpha=0.5,
273         )
274         self.cursor.set_active(True)
275 
276         self.iq = self.s.get_image_quality()
277         self.ui.MplWidget.canvas.ax.imshow(
278             self.iq,
279             cmap="gray",
280             extent=(0, self.dataset.nav_shape[0], self.dataset.nav_shape[1], 0),
281         )
282         self.ui.MplWidget.canvas.draw()
283 
284         # plot pattern from upper left corner
285         self.plot_signal(x, y)
286         self.current_x, self.current_y = x, y
287 
288         self.cid = self.ui.MplWidget.canvas.mpl_connect(
289             "button_press_event", lambda event: self.on_click_navigator(event)
290         )
```

```
291         self.hover_id = self.ui.MplWidget.canvas.mpl_connect(
292             "motion_notify_event",
293             lambda event: self.on_hover_navigator(event),
294         )
295 
296     def plot_signal(self, x_index, y_index):
297         pattern = self.dataset.ebsd
298         signal = pattern.data[y_index, x_index]
299         self.ui.MplWidgetPattern.vbl.setContentsMargins(0, 0, 0, 0)
300         self.ui.MplWidgetPattern.canvas.ax.clear()
301         self.ui.MplWidgetPattern.canvas.ax.axis(False)
302         self.ui.MplWidgetPattern.canvas.ax.imshow(signal, cmap="gray")
303         self.ui.MplWidgetPattern.canvas.draw()
304 
305         self.current_x, self.current_y = x_index, y_index
306 
307     def on_click_navigator(self, event):
308         try:
309             self.click_rect.remove()
310         except:
311             pass
312 
313         if event.inaxes:
314             x, y = floor(event.xdata), floor(event.ydata)
315             self.plot_signal(x, y)
316             self.addPattern(x, y)
317             self.update_rectangles()
318             self.current_rectangle(x,y)
319             self.ui.MplWidget.canvas.draw()
320 
321     def update_rectangles(self):
322         try:
323             for rect in self.clicked_patterns:
324                 rect.remove()
325         except:
326             pass
327         self.clicked_patterns = []
328         for coordinate in self.coordinates:
329             self.clicked_patterns.append(
330                 self.ui.MplWidget.canvas.ax.add_patch(
331                 Rectangle(
332                     coordinate,
333                     1,
334                     1,
335                     linewidth=0.5,
336                     edgecolor="black",
337                     facecolor="lime",
338                     alpha=1,
339                 )
340             ))
341 
342     def current_rectangle(self, x, y):
```

```
343         self.click_rect = self.ui.MplWidget.canvas.ax.add_patch(
344             Rectangle(
345                 (x, y),
346                 1,
347                 1,
348                 linewidth=0.5,
349                 edgecolor="black",
350                 facecolor="red",
351                 alpha=1,
352             )
353         )
354 
355     def on_hover_navigator(self, event):
356         if event.inaxes:
357             x_hover, y_hover = floor(event.xdata), floor(event.ydata)
358             self.ui.labelxy.setText(f"({x_hover}, {y_hover})")
359             self.plot_signal(x_hover,y_hover)
360         else:
361             self.ui.labelxy.setText(f"(x, y)")
362 
363 
364 ### PATTERN CENTER OPTIMIZATION ###
365 
366     def runPatternCenterOpimization(self):
367         """
368         Initializes patterCenterOptimization in a thread
369         """
370         basename, _ = path.basename(self.pattern_path).split(".")
371         save_dir = path.join(self.working_dir, f"{basename}_PC")
372         self.inlier_limit = float(self.ui.spinBoxInlier.value())
373 
374         sendToJobManager(
375             job_title=f"PC optimization {self.pattern_name}",
376             output_path=save_dir,
377             listview=self.parentWidget().ui.jobList,
378             func=self.patternCenterOpimization,
379             allow_cleanup=True,
380             allow_logging=True,
381         )
382 
383     def patternCenterOpimization(self):
384         print("Initializing pattern center optimization...\n\n")
385         basename, extension = path.basename(self.pattern_path).split(".")
386         save_dir = path.join(self.working_dir, f"{basename}_PC")
387 
388         try:
389             mkdir(save_dir)
390         except FileExistsError:
391             pass
392 
393         self.savefig_kw = dict(dpi=400, pad_inches=0, bbox_inches="tight", transparent=True)
394 
```

```
395         #set up master patterns, set up reciprocal lattice vector
396         simulator_dict = self.retrieveMPData()
397 
398         #draw positions on map
399         self.drawPCs(save_dir=save_dir)
400 
401         #Optimize PC, refine PC, create figures
402         self.optimizePC(simulator_dict, save_dir)
403 
404         #Overwite project settings file
405         self.save_project_settings()
406 
407         self.close()
408 
409         print("\nPattern center optimization complete.")
410 
411     def loadMP(self, name, mppath):
412         mp_i = kp.load(
413             path.join(mppath),
414             projection="lambert",
415             energy=self.energy,
416         )
417         mp_i.name = name
418         mp_i.phase.name = name
419         lat_i = mp_i.phase.structure.lattice
420         lat_i.setLatPar(10 * lat_i.a, 10 * lat_i.b, 10 * lat_i.c)
421         return mp_i
422 
423     def retrieveMPData(self):
424         """
425         Makes sure a maximum of one FCC and one BCC phase are used for optmization
426         """
427 
428         self.energy = self.s.metadata.Acquisition_instrument.SEM.beam_energy
429         self.mp_dict = {}
430 
431         FCCavailable, BCCavailable = True, True
432         for name, h5path in self.mp_paths.items():
433             mp = self.loadMP(name, h5path)
434             space_group = mp.phase.space_group.short_name
435             if space_group == ALLOWED_SPACE_GROUPS[0] and FCCavailable: # FCC
436                 self.mp_dict[name] = mp
437                 FCCavailable = False
438             elif space_group == ALLOWED_SPACE_GROUPS[1] and BCCavailable: # BCC
439                 self.mp_dict[name] = mp
440                 BCCavailable = False
441             elif space_group == ALLOWED_SPACE_GROUPS[0]: # FCC
442                 FCCavailable = False
443                 print("Hough indexing only supports one FCC/BCC phase. Using the first FCC phase for patter
                    ↪  center optimization.")
444             elif space_group == ALLOWED_SPACE_GROUPS[1]: # BCC
445                 BCCavailable = False
```

```
446                 print("Hough indexing only supports one FCC/BCC phase. Using the first BCC phase for patter
                    ↪  center optimization.")
447             else:
448                 print("Phase "+name+" is not supported with pattern center optimization.")
449 
450         ref_dict = {}
451         for name in self.mp_dict.keys():
452             rlv_i = ReciprocalLatticeVector.from_min_dspacing(self.mp_dict[name].phase)
453             rlv_i.sanitise_phase()
454             rlv_i = rlv_i.unique(use_symmetry=True)
455             rlv_i = rlv_i[rlv_i.allowed]
456             rlv_i.calculate_structure_factor()
457             structure_factor = abs(rlv_i.structure_factor)
458             order = np.argsort(structure_factor)[::-1]
459             rlv_i = rlv_i[order]
460             rlv_i = rlv_i[:4]
461             rlv_i.calculate_theta(self.energy * 1e3)
462             rlv_i = rlv_i.symmetrise()
463 
464             ref_dict[name] = rlv_i
465 
466         simulator_dict = {}
467         for name in self.mp_dict.keys():
468             simulator_dict[name] = kp.simulations.KikuchiPatternSimulator(
469                 ref_dict[name]
470             )
471 
472         return simulator_dict
473 
474     def drawPCs(self, save_dir):
475         """
476         Get the right format for coordinates
477         """
478         cr = np.array(self.coordinates).T
479         self.rc = cr[::-1]
480         self.s_cal = kp.signals.LazyEBSD(self.s.data.vindex[tuple(self.rc)])
481         self.s_cal.compute()
482 
483         out = da.compute(self.iq)
484         iq2 = out[0]
485         fig = kp.draw.plot_pattern_positions_in_map(
486             self.rc.T+0.5,
487             roi_shape = iq2.shape,
488             roi_image = iq2,
489             return_figure = True,
490         )
491         fig.savefig(path.join(save_dir, "maps_pc_cal_patterns.png"), **self.savefig_kw)
492 
493     def optimizePC(self, simulator_dict, save_dir):
494         """
495         Optimizes and refines pattern center. Creates figures for geometrical simulation, scatter plot and
    ↪  a txt file containing the results.
```

```
496
497         Parameters
498         ----------
499         simulator_dict : dict
500             Dictionary containing master pattern(s) of specified FCC and/or BCC phase.
501         save_dir : str
502             Directory where text file and pictures from optimization will be saved.
503         """
504         phase_list = PhaseList()
505         for mp in self.mp_dict.values():
506             phase_list.add(mp.phase)
507
508         print(phase_list)
509
510         self.updatePCArrayFromSpinBox()
511
512         det_cal0 = kp.detectors.EBSDDetector(
513             shape=self.s_cal.axes_manager.signal_shape[::-1],
514             sample_tilt=self.s_cal.detector.sample_tilt,  # Degrees
515             pc=self.pc,
516             convention=self.convention,
517         )
518         indexer_cal0 = det_cal0.get_indexer(phase_list, nBands=9)
519
520         if self.convention == "TSL": #Inital guess for hough_indexing_optimize_pc must be in BRUKER
        ↪  convention
521             self.pc = (self.pc[0], 1-self.pc[1], self.pc[2])
522
523         det_cal = self.s_cal.hough_indexing_optimize_pc(
524             self.pc,
525             indexer=indexer_cal0,
526             batch=True,
527         )
528
529         xmap_cal_ref_list, det_cal_ref_list = [], []
530
531         for mp_i in self.mp_dict.values():
532             indexer_cal = det_cal.get_indexer(PhaseList(mp_i.phase),
            ↪  nBands=indexer_cal0.bandDetectPlan.nBands)
533             xmap_cal = self.s_cal.hough_indexing(PhaseList(mp_i.phase), indexer_cal)
534
535             xmap_cal_ref, det_cal_ref = self.s_cal.refine_orientation_projection_center(
536                 xmap_cal,
537                 det_cal,
538                 master_pattern = mp_i,
539                 energy=self.energy,
540                 method="LN_NELDERMEAD",
541                 trust_region=[10, 10, 10, 0.2, 0.2, 0.2],
542                 rtol=1e-4,
543                     )
544             xmap_cal_ref_list.append(xmap_cal_ref)
545             det_cal_ref_list.append(det_cal_ref)
```

```
546
547         refined_xmap = kp.indexing.merge_crystal_maps(
548             xmap_cal_ref_list,
549             scores_prop="scores",
550         )
551
552         # generate figures
553         sim_i = simulator_dict.values()
554         sim_cal_ref_dict = {}
555         for phase, sim_i in simulator_dict.items():
556             sim_cal_ref = sim_i.on_detector(det_cal_ref,
                ↪  refined_xmap.rotations.reshape(*refined_xmap.shape))
557             sim_cal_ref_dict[phase] = sim_cal_ref
558
559         # geometrical simulations
560         fig, axes = plt.subplots(nrows=ceil(1*np.sqrt(len(self.coordinates)/2)),
            ↪  ncols=ceil(2*np.sqrt(len(self.coordinates)/2)), figsize=(10 * det_cal.aspect_ratio, 6))
561         for i, ax in enumerate(axes.ravel()):
562             if i >= len(self.coordinates):
563                 ax.axis("off")
564                 continue
565             ax.imshow(self.s_cal.data[i], cmap="gray")
566             phase = refined_xmap.phases[refined_xmap.phase_id[i]].name
567             lines = sim_cal_ref_dict[phase].as_collections(i)[0]
568             ax.add_collection(lines)
569             ax.axis("off")
570         fig.tight_layout()
571         fig.savefig(path.join(save_dir, "pc_geometrical_simulations.png"), **self.savefig_kw)
572
573         # scatterplot
574         fig = det_cal_ref.plot_pc("scatter", annotate=True, return_figure=True)
575         fig.savefig(path.join(save_dir, "pc_scatter_plot.png"), **self.savefig_kw)
576
577         # remove outliers
578         is_inlier = xmap_cal_ref.scores > self.inlier_limit
579         det_cal_ref.pc = det_cal_ref.pc[is_inlier]
580
581         # update pattern center
582         if not np.isnan(det_cal_ref.pc_average[0]):
583             self.pc = np.array(det_cal_ref.pc_average.round(4))
584
585         # Write to file
586         det_cal_ref.save(path.join(save_dir, "pc_optimization.txt"))
587
588         inliers, outliers = {}, {}
589         for i, coord in enumerate(self.coordinates):
590             if is_inlier[i]:
591                 inliers[i] = coord
592             else:
593                 outliers[i] = coord
594
595         f = open(path.join(save_dir, "pc_optimization.txt"), "a")
```

```
596         f.write("\n# Calibration patterns: "+str(self.coordinates))
597         f.write("\n# Inliers: "+str(inliers))
598         f.write("\n# Outliers: "+str(outliers))
599         f.write("\n# Mean pattern center: "+str(det_cal_ref.pc_average.round(4)))
600         f.write("\n# Standard deviation: "+str(abs(det_cal_ref.pc_flattened.std(0))))
601         f.close()
602 
603     def save_project_settings(self):
604         """
605         Saves average pattern center to project_setting.txt
606         """
607         self.setting_file.delete_all_entries()  # clean up initial dictionary
608 
609         ### Sample parameters
610         for i, mppath in enumerate(self.mp_paths.values(), 1):
611             self.setting_file.write(f"Master pattern {i}", mppath)
612 
613         self.setting_file.write("Convention", self.convention)
614         if self.convention == "TSL":
615             self.pc = (self.pc[0], round(1-self.pc[1], 4), self.pc[2])
616         self.setting_file.write("PC", f"{tuple(self.pc)}")
617 
618         self.setting_file.save()
```

Appendix B

Pattern Center Optimization: Method 2

The following Python code demonstrates Method 2 for optimizing pattern centers. This method utilizes a 6x6 grid consisting of ferrite and martensite patterns. The code is a conversion from a Jupyter notebook to Python, with each cell separated by "# %%", as presented in the python script.

```
# %%
# Exchange inline for notebook or qt5 (from pyqt) for interactive plotting
%matplotlib inline

import matplotlib.patches as mpatches
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

from diffsims.crystallography import ReciprocalLatticeVector
import hyperspy.api as hs
import kikuchipy as kp
from orix import plot
from orix.crystal_map import PhaseList

plt.rcParams.update(
    {
        "figure.facecolor": "w",
        "figure.dpi": 75,
        "figure.figsize": (8, 8),
        "font.size": 15,
    }
)

# %%
s = kp.load("/Users/olavlet/EBSD-data/DI-dataset/ferrite martensite/Pattern_test2.h5")
mp_m =
↪  kp.load("/Users/olavlet/EBSD-data/crystal_data/martensite_new/martensite05/martensite05_mc_mp_20kv.h5")
mp_f = kp.load("/Users/olavlet/EBSD-data/crystal_data/martensite_new/ferrite/ferrite_mc_mp_20kv.h5",
↪  projection="lambert")

mp_f.plot()
```

```

# %%
phase = mp_f.phase
phase.name = "ferrite"
lat = phase.structure.lattice
lat.setLatPar(lat.a * 10, lat.b * 10, lat.c * 10)

print(phase)
print(phase.structure)

# %%
grid_shape = (6, 6)
s_grid, idx = s.extract_grid(grid_shape, return_indices=True)
s_grid

# %%
iq = s.get_image_quality()

# For convenience, use the plot method of the crystal map attached to the EBSD
# signal to plot the IQ map. The array must be 1D.
s.xmap.scan_unit = "um"
fig = s.xmap.plot(
    iq.ravel(),
    vmax=0.3,
    remove_padding=True,
    colorbar=True,
    colorbar_label="IQ",
    return_figure=True,
)

kp.draw.plot_pattern_positions_in_map(
    rc=idx.reshape(2, -1).T,
    roi_shape=s.xmap.shape,
    axis=fig.axes[0],
    color="w",
)

# %%
signal_mask = ~kp.filters.Window("circular", s_grid.detector.shape).astype(
    bool
)

fig = plt.figure(figsize=(10,10))
_ = hs.plot.plot_images(
    s_grid * ~signal_mask,
    fig=fig,
    axes_decor=None,
    label=None,
    colorbar=False,
    per_row=6,
    padding=dict(wspace=0.03, hspace=0.03),
    tight_layout=True,
```

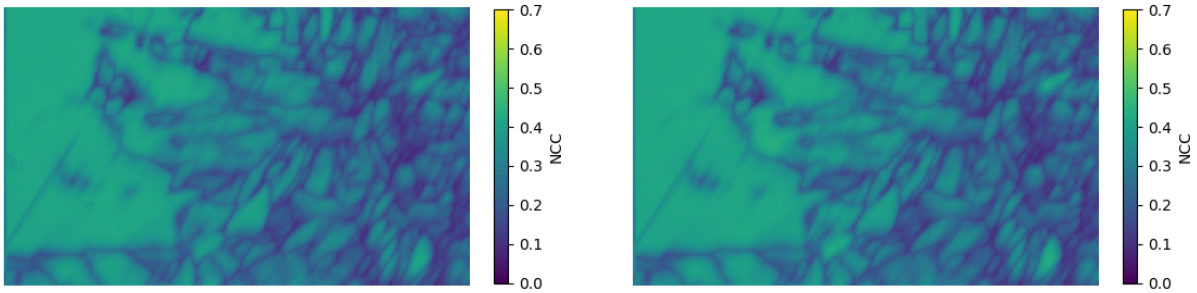
```
)
for i, ax in enumerate(fig.axes):
    ax.text(1, 1, i, va="top", ha="left", c="w")
    ax.axis("off")

# %%
det = s_grid.detector
det.pc = (0.556, 0.22, 0.584)
det

# %%
phase_list = PhaseList(phase)
phase_list

# %%
indexer = det.get_indexer(phase_list)
indexer.phaselist

# %%
det_grid = s_grid.hough_indexing_optimize_pc(
    pc0=det.pc,
    indexer=indexer,
    batch=True,
)

print(det_grid.pc_flattened.mean(axis=0))
print(det_grid.pc_flattened.std(0))

# %%
det_grid.plot_pc("scatter", annotate=True)

# %%
indexer = det_grid.get_indexer(phase_list)

# %%
xmap_grid = s_grid.hough_indexing(
    phase_list=phase_list, indexer=indexer, verbose=2
)

# %%
fig, axes = plt.subplots(ncols=2, figsize=(16, 4))
for ax, to_plot, label in zip(
    axes, ["fit", "cm"], ["Pattern fit [deg]", "Confidence metric"]
):
    im = ax.imshow(xmap_grid.get_map_data(to_plot))
    fig.colorbar(im, ax=ax, label=label, pad=0.02)
fig.subplots_adjust(wspace=.2)

# %%
xmap_grid_ref, det_grid_ref = s_grid.refine_orientation_projection_center(
    xmap=xmap_grid,
    detector=det_grid,
```

```
    master_pattern=mp_f,
    energy=20,
    method="LN_NELDERMEAD",
    trust_region=[1.0, 1.0, 1.0, 0.01, 0.01, 0.01],  # Wide trust region (!)
    rtol=1e-7,
    signal_mask=signal_mask,
    # Recommended when refining few patterns
    chunk_kwargs=dict(chunk_shape=1),
)

# %%
print(xmap_grid_ref.scores.mean())
print(xmap_grid_ref.num_evals.mean())
print(det_grid_ref.pc_average)
print(det_grid_ref.pc_flattened.std(axis=0))

# %%
det_grid_ref.plot_pc("scatter", annotate=True)

# %%
xmap_grid_ref.plot(
    "scores",
    remove_padding=True,
    colorbar=True,
    colorbar_label="NCC",
    figure_kwargs=dict(figsize=(4, 4)),
)

# %%
rlv = ReciprocalLatticeVector.from_min_dspacing(phase, 1)
rlv.sanitise_phase()  # Expand unit cell
rlv.calculate_structure_factor()
structure_factor = abs(rlv.structure_factor)
rlv = rlv[structure_factor > 0.5 * structure_factor.max()]
rlv.print_table()

# %%
simulator = kp.simulations.KikuchiPatternSimulator(rlv)

# %%
sim = simulator.on_detector(
    det_grid_ref, xmap_grid_ref.rotations.reshape(*xmap_grid_ref.shape)
)

# %%
fig, axes = plt.subplots(
    nrows=6, ncols=6, figsize=(16, 16)
)
for i, rc in enumerate(np.ndindex(grid_shape)):
    axes[rc].imshow(s_grid.data[rc] * ~signal_mask, cmap="gray")
    axes[rc].axis("off")
    lines = sim.as_collections(rc)[0]
```

```
187         axes[rc].add_collection(lines)
188         axes[rc].text(1, 1, i, va="top", ha="left", c="w")
189     fig.subplots_adjust(wspace=0.03, hspace=0.03)
190
191
```

Appendix C

Normalized Cross Correlation Maps



(a) **(b)**

Figure C.1: NCC Map corresponing to the phase maps in Figure 4.6, using Method 1 for PC optimization.

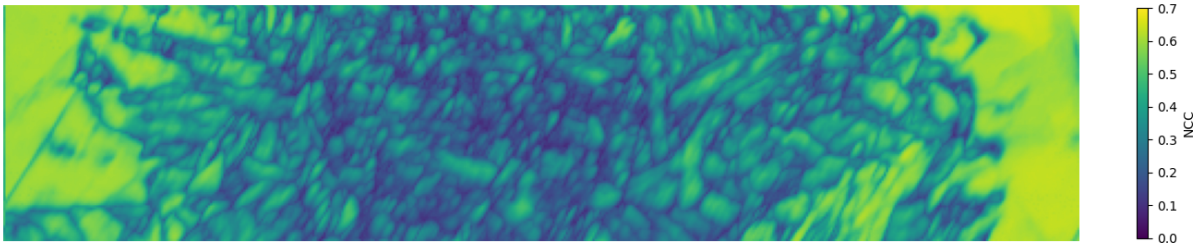


Figure C.2: NCC Map corresponing to the phase maps in Figure 4.7, using Method 2 for PC optimization.

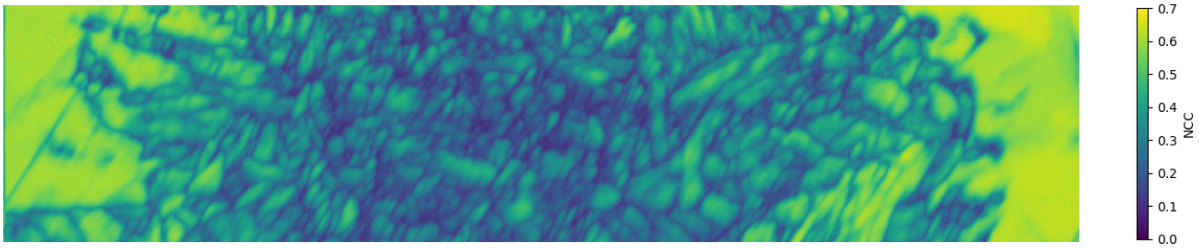


Figure C.3: NCC Map corresponing to the phase maps in Figure 4.8, using Method 3 for PC optimization without pseudo-symmetric operations.

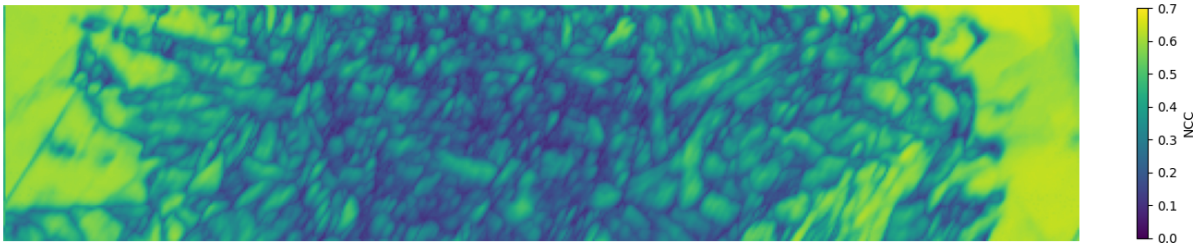


Figure C.4: NCC Map corresponing to the phase maps in Figure 4.9, using Method 3 for PC optimization with pseudo-symmetric operations.

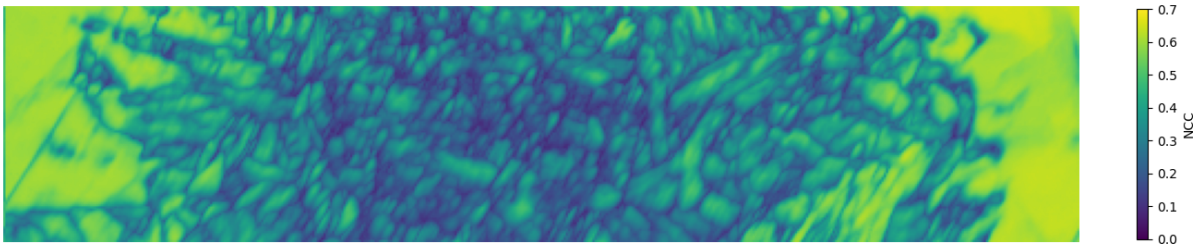


Figure C.5: NCC Map corresponing to the phase maps in Figure 4.10, using Method 3 for PC optimization with pseudo-symmetric operations.

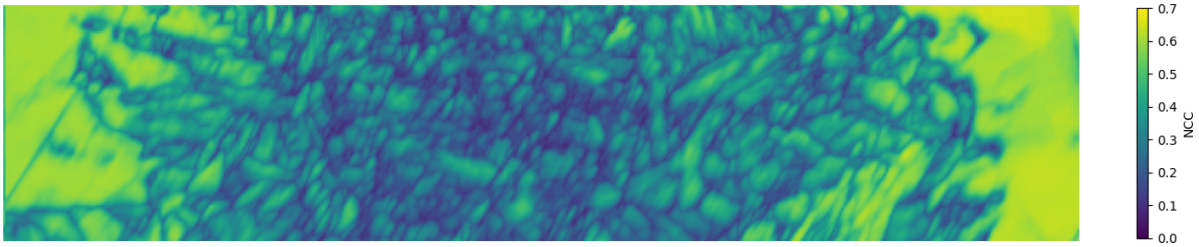
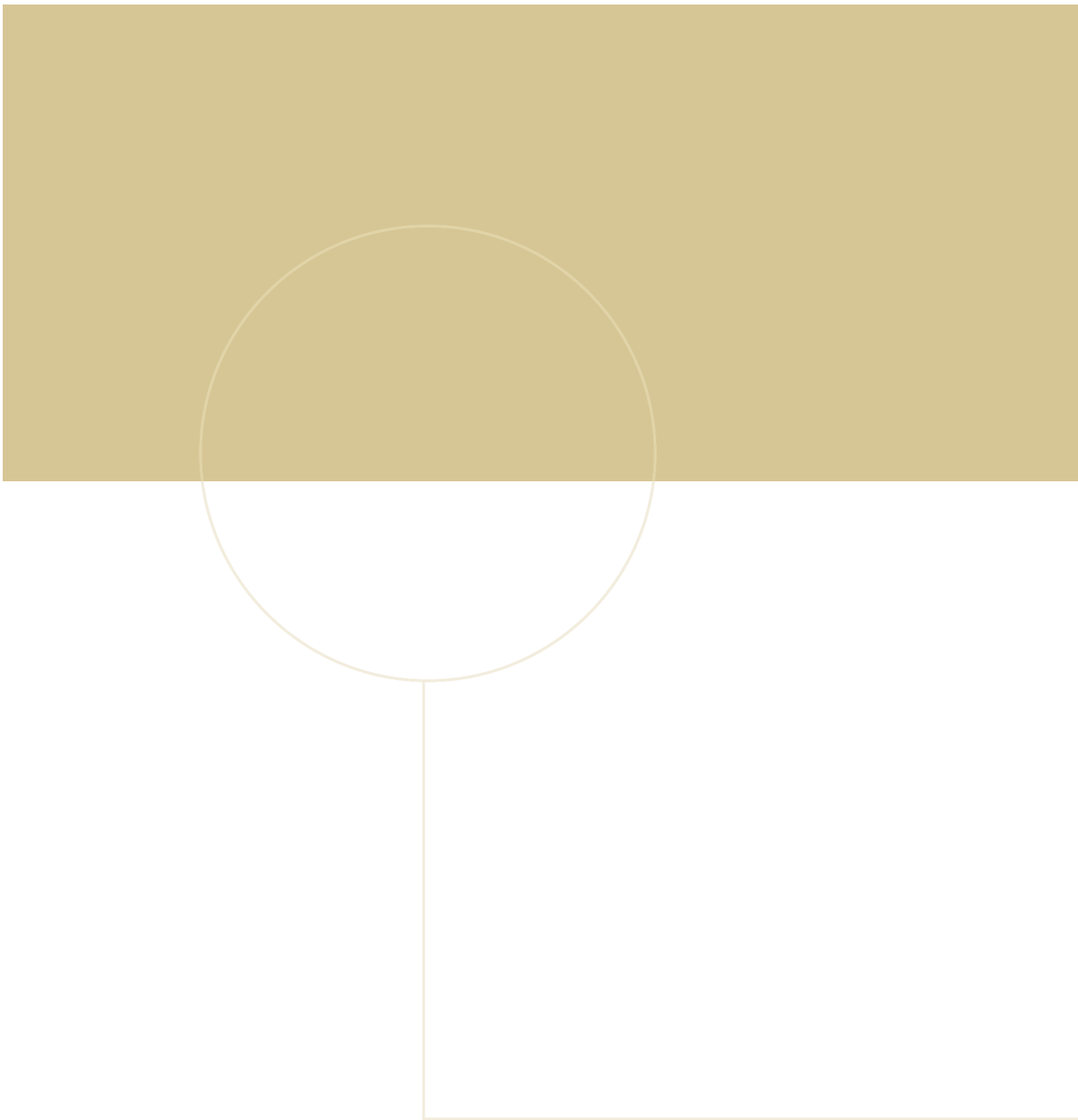


Figure C.6: NCC Map corresponing to the phase maps in Figure 4.11, using Method 4 (TSL) for PC optimization.



NTNU

Norwegian University of
Science and Technology