

Haakon Tvedt

AutomAl 6000: Semi-automatic structural labeling of HAADF-STEM images of precipitates in Al-Mg-Si-(Cu) alloys

Master's thesis in physics

Supervisor: Randi Holmestad, Calin D. Marioara

August 2020

Haakon Tvedt

**AutomAI 6000: Semi-automatic
structural labeling of HAADF-STEM
images of precipitates in Al-Mg-Si-(Cu)
alloys**

Master's thesis in physics

Supervisor: Randi Holmestad, Calin D. Marioara

August 2020

Norwegian University of Science and Technology



Norwegian University of
Science and Technology

Sammendrag

Når Al-Mg-Si (6xxx) legeringer utherdnes, får vi en rik samling av utfellinger eller presipitater, de fleste som nåler, stenger eller staver langs $\langle 100 \rangle$ retningene i aluminiumsgitteret. Disse presipitatene har betydelig innflytelse på materialets egenskaper, som for eksempel duktilitet og styrke, og det foregår mye forskning for bedre å forstå utfellingssekvensen til legeringer under utharding. Atomstrukturen i flere av fasene som dukker opp som presipitater i dette systemet kan beskrives som atomkolonner av legeringselementer (Mg, Si og Cu) med periodisitet langs $\langle 100 \rangle$ Al. Det har vist seg at disse kolonnene i 6xxx-systemet som oftest er karakterisert av et sett med enkle strukturelle prinsipper eller regler, som gjør det mulig å identifisere atomslag og den relative forskyvningen (langsgående hopp) av atomkolonnene som omfatter hele tverrsnittstrukturen på nålene. Med noen unntak har disse reglene vist seg å være nyttige, ikke bare for å analysere fasene i dette systemet, men også for å karakterisere uordnede presipitater, grenseflater og hybride utfellinger som inneholder flere faser.

Moderne forskning innen nanoskala materialvitenskap er utstyrt med stadig mer sofistikerte verktøy og teknikker, hvor atomopløsnings-aberrasjonskorrigert høyvinkel angulær mørkfelt-skanning transmisjonselektronmikroskopi (HAADF-STEM) spiller en viktig rolle. Med målet om å analysere slike HAADF-STEM-bilder av presipitattverrsnitt i Al-Mg-Si- (Cu) -systemet, har vi utviklet den frittstående programvaren AutomAl 6000.

AutomAl 6000 har blant annet en metode for kolonnedeteksjon, en algoritme basert på symbiosen til en statistisk modell og reglene som er formulert i et digraflignende rammeverk, samt et grafisk brukergrensesnitt. Fra et 2D-projisert HAADF-STEM-bilde av et tverrsnitt til et presipitat, kan programvaren semi-autonomt bestemme 3D-kolonneposisjoner og atomslag hvor reglene gjelder. I sin tur kan AutomAl 6000 deretter brukes til å vise atomoverlegg av utfellingen, analysere eller vise forskjellige andre representasjoner av dataene. Linker til programmet, så vel som andre ressurser, finnes på <http://automal.org>.

Abstract

The 6xxx Al alloy series, when subject to age hardening, can present a rich collection of precipitate structures, most extending as needles, laths or rods along the $\langle 100 \rangle$ directions in the aluminium matrix. These precipitates significantly influence properties of the bulk material, most notably ductility and strength, and are therefore the objective of extensive research aimed at better understanding the precipitation sequence of alloys during aging. The atomic structures of the majority of the phases that appear in this system, can be described as bundles of atomic columns following the matrix periodicity along $\langle 100 \rangle$ Al, where most are hijacked by the solute elements. It has been found that the columns in the 6xxx system precipitates are commonly characterized by a set of simple structural principles or rules, that allow identification of the species and the relative displacement (longitudinal jumps) of the atomic columns that comprise the entire cross-section structure of the needles. With some exceptions, these rules have proved to be useful for not only analyzing the phases in this system, but also to characterize disordered precipitates, matrix/precipitate interfaces, and hybrid precipitates containing multiple phases.

Modern research in the field of nano-scale material science is equipped with increasingly sensitive tools and techniques, of which atomic resolution aberration-corrected high-angle annular dark field scanning transmission electron microscopy (HAADF-STEM) plays an important part. With the goal of analyzing such HAADF-STEM images of particle cross-sections in the Al-Mg-Si-(Cu) system, we have developed the stand-alone software AutomAl 6000.

AutomAl 6000 features, amongst other, a method for column detection, an algorithm based on the symbiosis of a statistical model and the rules formulated in a digraph-like framework, as well as an approachable graphical user interface. From a 2D projected HAADF-STEM image of a cross-section precipitate, the software can semi-autonomously determine the 3D column positions and column species wherever the rules are applicable. In turn, AutomAl 6000 can then be used to display atomic overlays of the precipitates, analyze or display various other representations of the data. Links to the active repository, as well as other resources can be found at <http://automal.org>.

Preface and acknowledgments

This thesis is the conclusion of my MSc study in physics at NTNU. The central result of my work is represented by the software program AutomAl 6000¹. This thesis provides the theoretical and methodological background of the software. Some parts of the software has been challenging to describe in a written text, and the program should be tested and tried out. However, be advised that the software is not yet fully finalized at the time of writing, partly because this project has become very large for a one-man programming team, and time-restraints and many overrun deadlines has put some limits to what I was able to achieve in time, including the "fullness" of this thesis. It has also been a sometimes stressful challenge to manage the sheer size of this project, and its many aspects. Some parts of this thesis suffers a bit in quality as a consequence. I will however stay with NTNU for a bit longer in a small time-limited position, where my assignment will be to make the source code more finalized and ready to perhaps be adopted by the community as an open-source project, if the interest is there. We are also considering to write a paper to accompany the software, if time permits. With this in mind, accept my submission here as a snapshot of an still ongoing project.

I would like to thank my project supervisor, Randi Holmestad, Professor at the department of physics, NTNU, for approaching me with the subject for this project, and for the encouragements and assistance throughout the project. It has been very rewarding and motivating to work with the topics that have presented themselves through this project, and I am very grateful for the opportunity I was given, to be able to work with the subject matter, and to meet many of the people that make up the very competent community of TEM, aluminium and material science groups at NTNU and SINTEF.

I would also like to thank Calin Daniel Marioara, Senior Research Scientist at SINTEF industry, for spending his time teaching me some of the essential knowledge that I required to even *begin* approaching the challenges of this work. He has also provided feedback and assistance during the semester and inspired many of the implemented and proposed functionalities of the software associated with this work.

I would also like to mention Sigmund Jarle Andersen, Senior research scientist at SINTEF industry. His work, among others, on the structural principles has been paramount for the feasibility of the work that is presented in this thesis. He also made himself available for discussions, and has provided some interesting insights into some of the details that have exposed themselves through this work.

There seems to have been several persons who have conceived and thought about the idea to attempt to create the type of software that has been done here. Of those, I would especially like to mention Sigurd Wenner, researcher at SINTEF industry, whose PowerPoint-slides constituted the main source of inspiration for the definition of the problem at hand.

The work I have done on this project, is somewhat theoretical, or rather methodical, in nature, and I have spent very little time on experimental techniques. There are several individuals here at NTNU and SINTEF who have been helping me with their experimental insights and knowledge, images, feedback to the software and much more. This help has been invaluable to me. Of these individuals, I would especially like to mention Jonas Kristoffer Sunde, Adrian Lervik, Elisabeth Thronsen and

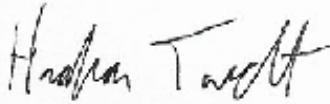
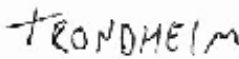
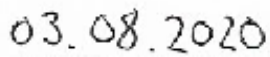
¹https://github.com/Haawk666/AutomAl_6000_thesis_version.git

Alexander B. Mosberg. I would also like to extend my gratitude to Christoph Hell who provided invaluable feedback on my writing style and some of the contents of this thesis.

In general I would also like to thank my family and friends, both near and far, who have supported me throughout my time at NTNU, and without whom I would not have been able to spend my time pursuing the inspired work that I have put into this project. My mother, Britt Tvedt, deserves particular mention for providing mental, economic and housing support. I would also like to thank my girlfriend, for her patience, support and love.

Let me also extend my gratitude to Kenneth Urdshals, who at a time when I had little to no inspiration in life, assisted and encouraged me to apply for funds that enabled me to acquire books and educational equipment which I needed to re-attempt passing high-school. Through a long chain of an increasing number of books and exams, this gesture has indirectly led to this thesis.

Finally, I wish to express my ever undying appreciation of my brothers of the ACNN. *Now rise and shine! O brethren of mine, And drink from these fountains of darkness.*

		
.....
Haakon Tvedt	Place	Date

Abbreviations

Al Aluminium. 4

API Application Programming Interface. 69

COM Center Of Mass. 14

CSV Comma Separated Values. 57

DM3 Digital Micrograph 3. 57, 69

FCC Face Centered Cubic. 4, 5

FFT Fast Fourier Transform. 14

GMS Gatan Microscopy Suite. 14

GUI Graphical User Interface. 1, 58, 59

HAADF-STEM High Angle Annular Dark Field Scanning Transmission Electron Microscopy. 1, 3

HTML HyperText Markup Language. 57

NN Nearest Neighbour. 5, 17, 25

SSSS Super Saturated Solute Solution. 5

SVG Scalable Vector Graphics. 57

Un Unknown. 17, 33, 35

Contents

Sammendrag	i
Abstract	ii
Preface and acknowledgments	iii
Abbreviations	v
1 Introduction	1
2 Theory	3
2.1 Brief introduction to HAADF-STEM imaging	3
2.2 Structural principles in Al-Mg-Si-(Cu) precipitates	4
2.2.1 Aluminium	4
2.2.2 Precipitation sequence and phase	5
2.2.3 Structural principles	5
2.3 Graph theory	7
2.3.1 Set-building notation	7
2.3.2 The general graph language	8
2.3.3 Digraphs	10
2.4 Data analysis and modelling	11
2.4.1 Normal distributions, Z-scores and normal probability plots	11
2.4.2 Multinomial multivariate normal distributions	12
3 Method	14
3.1 Column detection	14
3.1.1 Image pre-processing	14
3.1.2 Center of mass implementation	15
3.2 Atomic graphs	16
3.2.1 Basic atomic graph properties and vertex maps	16
3.2.2 Subgraph classifications	19
3.2.3 Numerical vertex attributes	21
3.2.4 Structural principles in atomic graphs	23
3.2.5 Graph operations	24
3.3 Statistical model for numerical vertex attributes	25
3.4 Column characterization	27
3.4.1 Sub-methods	27
3.4.2 Untangling	30
3.4.3 Composite algorithm	33
3.4.4 A detailed example	34

4	Discussion	48
4.1	Column detection	48
4.2	Atomic graphs	48
4.3	Column characterization	48
4.3.1	Statistical models	49
4.3.2	Zeta graphs	53
4.3.3	Algorithms versus machine learning	55
5	Conclusion	56
6	Further work	57
6.1	Technical docstrings	57
6.2	Exception protection	57
6.3	Additional export and import formats	57
6.4	Improvements to column detection	58
6.5	Improvements to column characterization	58
6.6	Optimizations	58
6.7	Improvements to the GUI	59
6.8	Web-page and bug-tracking	59
	References	60
	Glossary	63
A	Source code overview	69
A.1	Overview	69
A.2	Dependencies	69

1 Introduction

In many modern fields of research, analyzing data from experiments often requires the use of software. Applying general purpose software or writing simple special purpose programs is commonplace in many research projects, but sometimes a more substantial effort on development of analytical tools can be warranted. This thesis outlines such an effort, where a new stand-alone, open-source software tool entitled *AutomAl 6000* has been developed. AutomAl 6000 is a specialized tool for analyzing atomic resolution High Angle Annular Dark Field Scanning Transmission Electron Microscopy (HAADF-STEM) images of Al-Mg-Si-(Cu) precipitates in 6xxx aluminium alloys. From an input HAADF-STEM image, AutomAl 6000 can determine, given some manual input, the 3-d positions and atomic species of each atomic column in the image, which can be visualized with an *atomic overlay*. This overall process is possible due to the structural principles that accompany the precipitation sequences in these alloy systems, see section 2.2.3. The extraction of atomic structure data from HAADF-STEM images by using the methods herein, involves casting all the useful information in a suitable framework which have been termed *atomic graphs*, see section 3.2. Having the atomic data ordered in this manner, also lends itself naturally to additional novel ways of quantitative analysis of precipitates in terms of graph parameters, see section 4.

To successfully perform the tasks that AutomAl 6000 either claims or desires to include in its capabilities, there are many challenges that must be approached, resulting in an inherently multi-faceted project. The problems that are most deserving of highlight, are summarized below.

- **Column detection.** Before structural analysis can take place, a method that can automatically locate the atomic columns in HAADF-STEM images, is required.
- **Atomic graphs.** A mathematical framework for both the structural principles observed in most Al-Mg-Si-(Cu) precipitation structures, and a flexible statistical model. Atomic graphs also serves as a somewhat intuitive visual interface between a user of AutomAl 6000 and the state or structure of the underlying data.
- **Statistical modelling.** A basic framework for developing statistical models used in the methods of AutomAl 6000.
- **Column characterization.** Using the atomic graph framework together with a statistical model, an algorithm has been developed, that can solve the over-arching problem of transforming 2-dimensional column positions in the image plane, to 3-dimensional column positions, as well as the atomic species of the columns.
- **Graphical user interface and other secondary methods and tools.** To make the methods we have developed easily accessible to everyone, we have made a Graphical User Interface (GUI). There are also several methods and tools that are natural extensions to the software that have been implemented.

Due to the somewhat unusual format of this Masters's project, the output generated during the project period consists of the following several parts, which should all be considered together, with emphasis on this thesis and the source code:

- **This thesis**, which is focused on developing some amount of rigour and suitable language to systematically explain the thinking that has gone in to the development of the methods of AutomAl 6000.
- **A webpage**² that contains software downloads, guides and technical documentation. As well as some select updates on the project activities.
- **A git repository**³ with the source code of AutomAl 6000. The particular link given here is directed to a repository that is intended as a snap-shot of the source code at the time this thesis is submitted for consideration. In the near future, the active repository can be found under the "links" section of the webpage.

It should be noted, that with AutomAl 6000 being a piece of software that is still under development and constantly subject to change, this thesis is unlikely to give an accurate description of the software for very long⁴. Thus, implementation details are in general avoided in this text. Instead, the goal of this text is to present a language of methodology and to substantiate the thinking that has emerged from the process that has lead to the current version at the time of writing.

The theory section, section 2, of this thesis presents some basic introduction to HAADF-STEM experimental technique, aluminium alloys and the structural principles observed in the 6xxx precipitation sequence. Furthermore, the theory section also provides a brief introduction to the language of graph theory, as well as a short discussion of data modelling techniques and nomenclature. The several facets of the methodology of AutomAl 6000 is then presented in the method section, section 3. Some discussions arise naturally from several of the topics related to the development of AutomAl 6000, which are presented in section 4. In the conclusion section, section 5, the experience acquired during this project is summarized. The final section, section 6, presents the most pressing focus points as the project moves into its final stage. Appendix A gives an overview of the source code of AutomAl 6000.

There are no experimental results produced by this project, and the images presented in this thesis which AutomAl 6000 has been developed with, are borrowed from individuals at the department of physics at NTNU and SINTEF industry. These images are presented here with permission.

²<http://automal.org>

³https://github.com/Haawk666/AutomAl_6000_thesis_version.git

⁴This thesis might be somewhat outdated just weeks after submission, if the previous 2 years of development is anything to go by.

2 Theory

This section provides a brief insight into the topics that underpins the AutomAl 6000 software. After a short introduction to HAADF-STEM, a focused look into aluminium precipitates follow. Next, the language of graph theory is introduced. Finally, this sections ends with an overview of data analysis and statistics.

2.1 Brief introduction to HAADF-STEM imaging

Theoretical and experimental details of HAADF-STEM are not paramount to the understanding of the methods of this thesis. However, since these images are the basis of all the analysis done, a brief and simplistic outline is given here. More details can be found elsewhere [1].

In STEM mode, the incident electron beam is focused into a very sharp, convergent beam which is scanned across the surface of a very thin sample, typically less than 50 Å. To obtain high-resolution images, the sample has to be oriented exactly in a zone axis, which means that atoms are aligned in columns exactly along the beam direction. Furthermore, the probe size has to be less than the distance between the atomic columns. To obtain such a small probe, the microscope has to be corrected for spherical aberrations in the probe forming lens. The microscope used to obtain the images analyzed in this work, is the JEOL ARM 200, with a probe size <1 Å at 200 kV high voltage.

HAADF is a special imaging mode which collects transmitted electrons that are scattered to high angles through Rutherford and thermal diffuse scattering. Put simply (and almost correct), the electron scattering to high angles is inelastic and incoherent, while Bragg scattering to lower angles is elastic and coherent [1]. Typical collection angles for the HAADF detector in the back focal plane are from 40 to 200 mrad. How much of the incident electrons that is scattered to the detector, is approximately proportional to the square of the atomic number of the species in the column that is being hit by the beam and how close to the column the beam is. This intensity, which is the number of electrons scattered to the detector, is often referred to as *Z-contrast*. Each pixel in a HAADF-STEM image is a representation of the scattering measured from that point of the sample. The scan then proceeds to an adjacent area of the sample, and so on, until the entire image is produced. Figure 2.1 shows a simple schematic of the HAADF-STEM setup.

There are many experimental challenges that one must face to produce good images. As mentioned, to obtain atomic resolution, aberration correction is required. Furthermore, minute vibrations can cause the sample to drift slightly during the scan, causing a characteristic type of image distortion. To achieve images that show clear atomic columns, it is also important that the TEM-operator adjusts the sample such that the beam direction is perfectly aligned with the main zone axis. In the images analyzed in this thesis, we always study the $\langle 001 \rangle$ zone axis of the aluminium matrix. Physical properties of the specimen that are being studied, can also sometimes influence the visual results, one example being columns, structures or even precipitates that doesn't fully penetrate the sample. Finally, high quality samples⁵ are important for good results.

⁵In this context, high quality usually implies a thickness <50 Å, no carbon contamination and a very thin oxide surface layer.

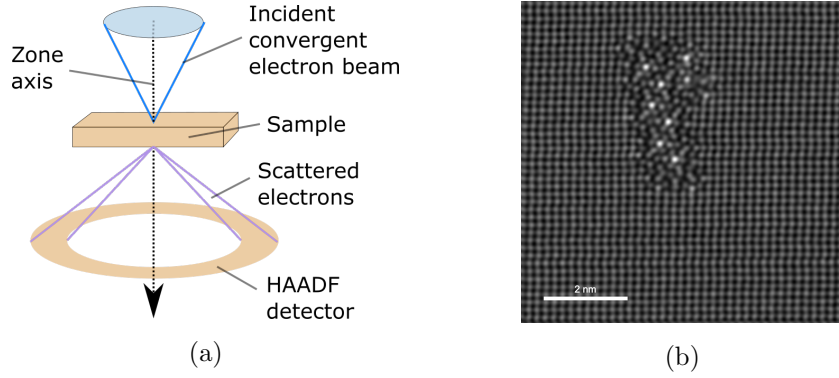


Fig. 2.1: HAADF-STEM micrograph. (a) A simple schematic of the HAADF-STEM setup, based on figure 1 in [2]. (b) An example of an atomic resolution image of an aluminium precipitate, acquired with JEOL ARM 200.

2.2 Structural principles in Al-Mg-Si-(Cu) precipitates

This section provides a brief insight into the landscape of aluminium precipitation. The following assumes some very basic familiarity with crystallography nomenclature, like *crystal lattice*, *crystal directions* or *Miller indices* [3].

2.2.1 Aluminium

Aluminium (Al) has a crystalline Face Centered Cubic (FCC) structure with lattice constant $a = 404.95$ pm, see figure 2.2 (a). In a HAADF-STEM image, we are imaging the structure along one of the equivalent $\langle 001 \rangle$ directions, which means that we see the projection of the FCC and atomic positions appear as closest neighbours in the projection in fact have a $\frac{1}{2}a$ component in the beam direction. In figure 2.2 (b) this projection is illustrated. Since the samples that are examined with atomic resolution STEM typically are 20-50 atomic layers thick, the atomic positions seen in the images, are referred to as *columns*.

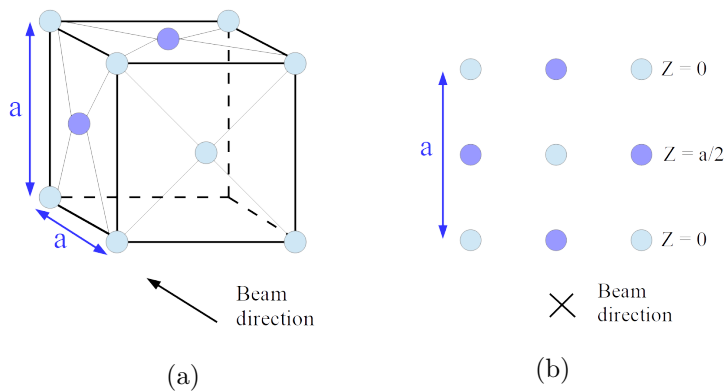
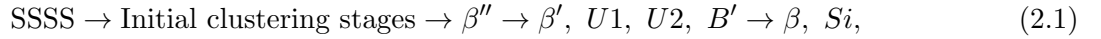


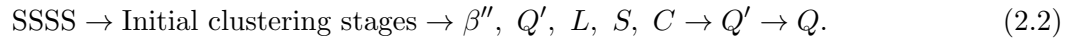
Fig. 2.2: The FCC aluminium structure with lattice constant a . (a) Seen at an angle. Atoms of same color occupy the same plane normal to the beam direction. For clarity, only atoms on the surfaces that are facing us are shown, but the cell is identical on all sides. (b) Projection along the beam direction. Z is the displacement in the beam direction.

2.2.2 Precipitation sequence and phase

When an alloying element is introduced into the FCC structure, it will influence how the material changes during thermomechanical treatment processes. Depending on the treatment, the alloying elements will often start to cluster together into molecule-like structures. This initial clustering then develops into precipitates. The intermediate stable or meta-stable phases that appear as precipitates during thermomechanical processing are often seen in the context of a precipitation sequence. With arrows representing some thermomechanical process, the Al-Mg-Si sequence is often written as



where SSSS is the *super saturated solute solution*, where the alloy elements are homogeneously mixed in the aluminium matrix, occupying aluminium FCC lattice points. The stable and meta-stable phases are named β'' , β' , $U1$ and so on, and they are characterized by the structure of the precipitate. These precipitates extend like needles, laths or rods along the $\langle 001 \rangle$ directions in the Al matrix [4]. Another precipitation sequence transpires in the presence of Cu:



The "library" of know stable and meta-stable phases is not necessarily exhaustive, and there are currently new phases still being described. It also stands to mention that sometimes, certain areas within a precipitate is of no particular phase, but rather takes on a disordered phase. Particles that contain the structures of two or several different phases is a *hybrid precipitate*. Some examples of familiar phases that appear in the 6xxx Al-Mg-Si-(Cu) systems [5], are shown in figure 2.3.

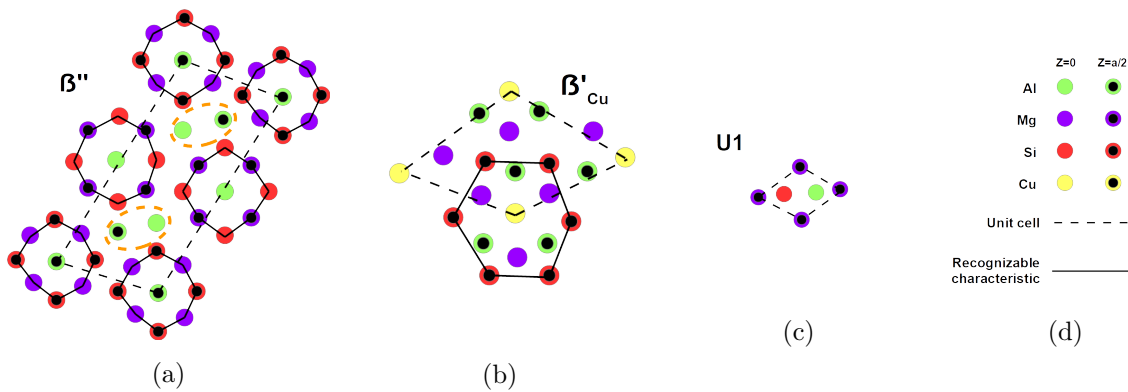


Fig. 2.3: A selection of example phases. (a) β'' -phase. The recognizable structure is nicknamed an 'eye'. (b) β'_{Cu} -phase. The recognizable structure is nicknamed 'star'. (c) $U1$ -phase. (d) Legend.

2.2.3 Structural principles

The current understanding of how the particular precipitation structures occur, is explained with solute elements with larger atomic radii accepting more Nearest Neighbours (NNs), while smaller atoms accept fewer [6]. In pure Al FCC, each Al position in the Al matrix have 12 NNs that are

separated by $\frac{1}{\sqrt{2}}a$. Taking a plane through the Al position and 4 of the NNs, there are 4 NNs in the same plane, and 4 NNs to each side of the plane. Seen in projection then, there are 4 NNs which occupy the opposite plane, see figure 2.4a, where the opposite plane NNs are indicated by white lines. Let the *symmetry* of each column be the number of opposite plane NNs in projection. Now, introduce a line defect by shifting one Al column into the *interstitial* position, $\frac{1}{2}a$ along a $\langle 001 \rangle$ direction. In figure 2.4b, this is illustrated with the central column now occupying the $Z = \frac{a}{2}$ plane. With this central column in the interstitial position, the symmetry of the surrounding columns change. Now let the columns that have changed symmetry be occupied by solute elements, assuming that columns with a 3-symmetry are favoured by the smaller atoms (Si, Cu), 4-symmetry by Al, and 5-symmetry by the larger atoms (Mg). This is illustrated in figure 2.4c. Finally, let the structure relax to compensate for the new atomic radii, which produces the familiar β'' eye, as seen in figure 2.4d.

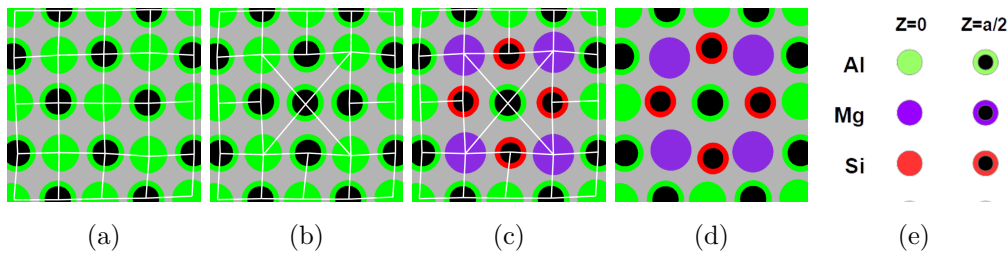


Fig. 2.4: Building the β'' eye from a line defect. White lines indicate the opposite plane NNs. (a) Projection of the Al FCC structure. (b) FCC with a line defect (into the page) at the central column. The symmetries of the surrounding columns change. (c) Let solute elements occupy the altered symmetry columns. (d) The recognizable characteristic of β'' as the column positions are shifted to accommodate the new atomic radii. (e) Legend.

This designation of atomic species by column symmetry, can be seen as a structural principle in the Al-Mg-Si-(Cu) system. All though the principle has several counterexamples in the system, it has proven useful and insightful for the analysis of precipitates. Not only does the principle hold for most of the familiar phases⁶ in Al-Mg-Si-(Cu), but it often also holds in highly disordered precipitates, as well as hybrid precipitates consisting of two or more phases that are combined together. Even the interfaces that appear between phases and the Al matrix, will typically adhere to this structural principle.

By using this principle, together with their experience of the phase library of the system, researchers can overlay HAADF-STEM images of precipitates, with column species and Z -height indicators, often without the need for other information than the image itself. Note that a single column might contain several different atomic species, but this model attempts to capture the majority species, as well as structural function. In this scheme then, Al matrix columns that are enriched with Cu for instance, would still be considered an Al column, despite the bright signature of Cu. As will be discussed in section 3.2.4, it is this principle AutomAl 6000 has put into a systematic framework, which enables it to overlay HAADF-STEM images.

⁶The most notable exceptions being β' , Q and $U1$.

2.3 Graph theory

Graph theory provides an elegant framework that can be used to solve many different types of problems, and can sometimes even reveal connections between problems that otherwise might have seemed disjoint. This section provides a short introduction to some of the basic concepts and definitions from graph theory that will be useful later. Most of the material presented here are based loosely on the introductory level textbook "Graphs and Digraphs" by Chartrand *et. al.* [7], which is only one of many extensive texts on the subject, see for instance [8]. Some of the notational conventions are slightly altered to better fit the style that are deploy in this text. To quickly cite the many definitions of graph theory can make for a very "dense" read. To combat this, the contents of this section is divided into small titled paragraphs, so as to facilitate both reading and back-referencing.

2.3.1 Set-building notation

Graph theory takes advantage of the relational language of set theory. This is a vast topic, but is presented here in its bare brevity and strictly limited to the notation that will be deployed.

Sets are collections of unique objects. If A is a set of a certain type of objects, and a is an object of that type, then a is either in A or not in A . *Membership* is notated as $a \in A$, if a is in A , and $a \notin A$, if a is not in A . The *cardinality* of a set A , is the number of elements in A , notated $|A|$. The *union* of two sets A and B is a new set C that contains all the objects that are in A or B or both, notated $C = A \cup B$. The *intersection* of two sets A and B is a new set C that contains all the objects that are in both A and B , notated $C = A \cap B$. The *disjunctive union* (sometimes termed the *symmetric difference*) of two sets A and B is a new set C that contains all the objects that are in A or B , but not in both, notated $C = A \oplus B$. The *difference* between two sets A and B , is a new set C that contains all the elements that are in A , but not in B , notated $C = A - B$. Table?

Many of the relationships between sets can be described efficiently with *set-builder notation* [9]. The idea is to define a range of objects, and each of those objects that satisfy a certain rule or equation, is included in a set. The notation uses curly brackets around the range and the rule, separated by the "|" delimiter, so as to signify that it is a set, like so: $\text{set} = \{\text{range}|\text{rule}\}$. As an example, the definition of the intersection of two sets A and B , can be defined using set-building notation as $A \cap B = \{\forall x \in (A \cup B) | x \notin (A \oplus B)\}$, which can be read as *for all x in the union of A and B that are not in the disjunctive union of A and B* . The symbol " \forall ", which reads "for all", can also be omitted here, because it is implied with this notation. Another example, defining the difference between A and B , is $A - B = \{x \in A | x \notin B\}$.

Set-building notation is especially useful in conjunction with programming implementations, because it is easy to translate into code. An illustration of this with python, again using the difference between two sets, is shown in listing 1. From this understanding as the set-building notation as a sort of pseudo code, we see that the range can be any kind of *iterable*, and the rule can be any test that always evaluates as true or false on that range, but since the curly brackets indicate a set, the notation will always return a set. If an object x is found more than once in the range, then x will still only occur once in C , because both mathematical sets and python sets disallow duplicated objects, as they should.

```

1 def set_difference(A, B):
2     """Take the difference between the sets A and B."""
3     C = set() # Instantiate an empty set
4     for x in A: # For all x in A,
5         if not x in B: # if x is not in B,
6             C.add(x) # then include x in C if x is not already in C.
7     return C

```

Listing 1: Example implementation of $\{x \in A | x \notin B\}$ with python.

2.3.2 The general graph language

Vertices

A *vertex* v_i , or *vertices* in plural, is thought of as a point or node, and is the fundamental building block of a graph. In the visual representation of graphs, vertices are often represented by points or dots, but note that vertices have no spatial location, and are only located in a graph through their connections with other vertices.

Arcs

An *arc* a_j is a set of two vertices, indicating a relationship between those vertices. In *directed* arcs, the 2-element vertex subset is ordered.

Graphs as sets

A *graph* G , is a finite non-empty set V of vertices together with with a possibly empty set A of 2-element arc subsets of V . Thus $G = (V, A)$, where $V = \{v_1, v_2, \dots, v_n\} = \{v_i\}$ and $A = \{a_1, a_2, \dots, a_m\} = \{a_j\}$, where $a_j = v_k v_l$, given that v_k and v_l are elements of V . The number of vertices n of a graph is called its *order*, and the number of arcs m its *size*. The sets V and A are sometimes referred to as $V(G)$ and $A(G)$, so as to clarify the parent graph⁷. Figure 2.5 shows a simple example of a graph.

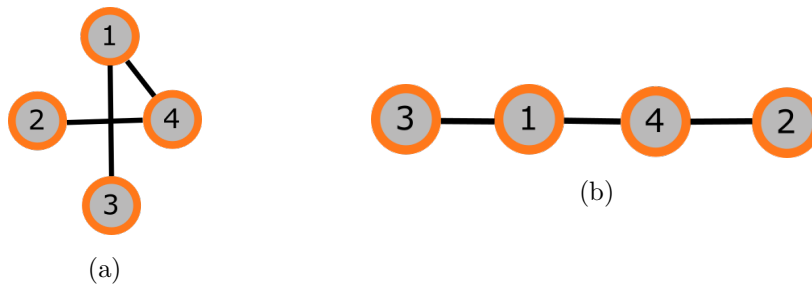


Fig. 2.5: (a) Visualization of a graph G with $V(G) = \{v_1, v_2, v_3, v_4\}$ and $A(G) = \{v_1v_4, v_1v_3, v_2v_4\}$. This graph has order $n = |V(G)| = 4$ and size $m = |A(G)| = 3$. (b) The same graph G with rearranged vertices. This is the same graph because it has the same vertex and arc set, and because in general, vertices do not have meaningful spatial positions.

⁷Useful when working with multiple graphs and/or sub-graphs

Adjacency, neighbourhoods, vertex degree and the first theorem of graph theory

If two distinct arcs in A have a common vertex, then they are called *adjacent arcs*. If $v_k v_l$ is an arc in $A(G)$, then v_k and v_l are *adjacent vertices*. Two adjacent vertices are *neighbours* of each other. The set of neighbours of a vertex v_i is called the (open) *neighbourhood* of v_i and is denoted $N(v_i)$. The closed neighbourhood of v_i also includes v_i itself and is denoted by $N[v_i]$. The number of vertices in the neighbourhood of v_i is called the *vertex degree* of v_i , and is denoted $\deg(v_i) = |N(v_i)|$. The maximum and minimum vertex degree in a graph G is denoted $\Delta(G)$ and $\delta(G)$ respectively. In an un-directed graph, the first theorem of graph theory^{8,9}, states that if G is a graph of size m and order n , then

$$\sum_{i=1}^n \deg(v_i) = 2m. \quad (2.3)$$

Another interesting graph parameter that we will mention here, is the *average degree* of G . In an un-directed graph, it follows from the first theorem of graph theory that

$$\frac{1}{n} \sum_{i=1}^n \deg(v_i) = \frac{2m}{n}. \quad (2.4)$$

Connectedness

Two vertices v_k and v_l are said to be *connected* if there exists a $v_k - v_l$ path P in G . A graph where every two vertices are connected, is a *connected graph*. Note the fact that two connected vertices are not necessarily adjacent.

Subgraphs

A graph H is a *subgraph* of G if $V(H) \subseteq V(G)$ and $A(H) \subseteq A(G)$, that is, $H \subseteq G$. If H is a subgraph of G , then G is also a supergraph of H . An *induced subgraph* H of G , is formed by a vertex subset $V(H) \subseteq V(G)$, together with the arc subset such that every arc in $A(G)$ where both vertices are in $V(H)$, are included in $A(H)$. Using set-builder notation, and given a vertex subset $V(H) \subseteq V(G)$, the induced subgraph H is $H = (V(H), A(H))$, where

$$A(H) = \{v_j v_k \in A(G) | v_j \in V(H) \wedge v_k \in V(H)\}. \quad (2.5)$$

Graph operations

Graph operations can be sorted into two classes, *unary operations*, which will create a new graph from a graph, and *binary operations*, which will create a new graph from two distinct graphs. This work is mostly concerned with unary operations. *Elementary operations*¹⁰ are unary operations that make small local changes to a graph, like deleting or adding an arc. Some operation definitions which are specific to this thesis, are given in section 3.2.5.

Classes of graphs

There are many ways to classify graphs. Only the most elementary classifications are defined here.

⁸Attributed to Leonhard Euler

⁹Theorem 1.4 in [7]

¹⁰Also termed *graph edits*.

Thus far, the discussion has implicitly been concerned with *undirected graphs*. That is, the arcs have no direction, ie $a_j = v_k v_l = v_l v_k$. In contrast to un-directed graphs, in a *directed graph*, also called a *digraph*, the arcs have a specific direction, ie $a_j = v_k v_l \neq v_l v_k$. Digraphs are discussed in more detail in the next section. Other classes include *multigraphs*, which are graphs which can contain more than one identical arc. *Pseudographs* are graphs that allow *loops*, that is, arcs that connect a vertex to itself, ie $a_j = v_k v_k$. Multigraphs and pseudographs are sometimes considered as sub-classes of digraphs. Pure digraphs that don't allow repeated arcs or loops are thus sometimes labelled *simple digraphs* for clarity. In this text the word "digraph" implies "simple digraph".

The adjacency matrix representation of graphs

There is a way to represent graphs in a binary matrix form. Termed the *adjacency matrix* of G , it is the $n \times n$ matrix M with binary valued elements

$$m_{ij} = \begin{cases} 1 & \text{if } v_i v_j \in A(G) \\ 0 & \text{if } v_i v_j \notin A(G). \end{cases} \quad (2.6)$$

Excluding pseudographs, the diagonal elements are always 0, since a vertex can not be adjacent to itself. Furthermore, for an un-directed graph, M is symmetric about the diagonal, but not for a digraph. Another useful property of M is that the sum of the elements in row i (or column i) gives the degree of vertex v_i . The study of the properties of the adjacency matrix forms the basis for *spectral graph theory*.

2.3.3 Digraphs

Using some of the fundamentals of graph theory, this section proceeds to discover the nomenclature of digraphs, which is similar to that of un-directed graphs.

A digraph G is a finite set of vertices $\{v_i\}$ together with a possibly empty set of ordered pairs of distinct vertices $\{a_j\}$ of G . In similar fashion as with un-directed graphs, one writes $G = (V(G), A(G))$, where $V(G) = \{v_1, v_2, \dots, v_n\} = \{v_i\}$ and $A(G) = \{a_1, a_2, \dots, a_m\} = \{a_j\}$, where $a_j = v_k v_l \neq v_l v_k$, given that v_k and v_l are elements of $V(G)$. As with un-directed graphs, the order n of a digraph G is the number of vertices in the vertex set, $n = |V(G)|$, and the size m of a digraph G is the number of arcs in the arc set, $m = |A(G)|$.

If $A(G)$ contains the arc $a_j = v_k v_l$, then v_k is said to be *adjacent to* v_l , and that v_l is *adjacent from* v_k . Alternatively, v_k is a *direct predecessor* of v_l , and v_l is a *direct successor* of v_k . This text will stick with the terms "adjacent to/from". The arc $v_l v_k$ is the *inverse arc* of $v_k v_l$ and vice versa.

The set of all vertices that are adjacent *from* v_i , is the *out-neighbourhood* of v_i , notated $N^+(v_i)$. The *out-degree* of v_i is then $\deg^+(v_i) = |N^+(v_i)|$. Similarly, the set of all vertices that are adjacent to v_i , is the *in-neighbourhood* of v_i , notated $N^-(v_i)$. The *in-degree* of v_i is then $\deg^-(v_i) = |N^-(v_i)|$. The first theorem of digraph theory now reads

$$\sum_{i=1}^n \deg^+(v_i) = \sum_{i=1}^n \deg^-(v_i) = m. \quad (2.7)$$

A digraph G , is a *symmetric digraph* if and only if, for every arc a_j in $A(G)$, $A(G)$ also contains the inverse of a_j . The adjacency matrix of a symmetric digraph is symmetric about the diagonal. As is explored further in section 3.4, transforming a simple digraph into a symmetric digraph, by applying simple elementary graph operations under certain restrictions, is the central moment of this work. Figure 2.6 shows a simple example of a digraph, illustrating some of the definitions given here.

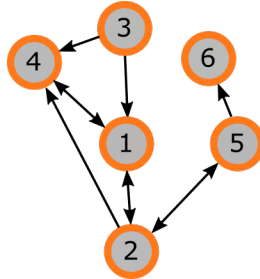


Fig. 2.6: A digraph $G = (V, A)$ of order $n = 6$ and size $m = 10$, where $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, and $A = \{v_1v_2, v_2v_1, v_1v_4, v_4v_1, v_2v_4, v_5v_2, v_2v_5, v_3v_4, v_3v_1, v_5v_6\}$. The out-degree of vertex v_6 , is $\deg^+(v_6) = 0$. A vertex with out-degree equal to 0, is called a sink, while a vertex with in-degree equal to 0, is called a source. Vertex v_3 is an example of a source. To illustrate another concept, the out-neighbourhood of vertex v_1 , is $N^+(v_1) = \{v_2, v_4\}$, which means that the out-degree of v_1 is $\deg^+(v_1) = |N^+(v_1)| = 2$.

2.4 Data analysis and modelling

In a very general sense, data can be said to have *nominal attributes* (categorical data), or *numerical attributes* (interval or ratio data). If we were to model an atom, we could for instance make a very simple model where every atom has a species, radius and number of electrons. In this example, the species of the atom (as defined by the number of protons in the core) is an example of a nominal attribute¹¹. Nominal data is discreet and typically within a preset limited range of possibilities. The radius of the atom is an example of a continuous numerical attribute, and the number of electrons is an example of a discreet numerical attribute. The data format which appears in this text, will typically have n datapoints, each with k numerical attributes and 1 nominal attribute. The concepts that are discussed in this chapter can be explored in more depth in [10, 11].

2.4.1 Normal distributions, Z-scores and normal probability plots

The well-known normal distribution is remarkably effective at capturing the essence of a wide range of natural distributions. Given a real-valued random variable x , with mean μ , and standard deviation σ , the normal distribution f takes the form of the continuous probability density function

$$f(x | \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp \left[-\frac{(x - \mu)^2}{2\sigma^2} \right]. \quad (2.8)$$

¹¹The atomic number can be used to represent atomic species in an ordinal hierarchy, but this indexing is still nominal. Let us say the range of the data was not the periodic table, but exclusively Mg and Si. The atomic numbers are then arbitrary in the way of categorizing *Mg* and *Si*.

The square of the standard deviation, σ^2 , is known as the variance of the distribution. For a given set of n data points $\{x_h\}$, the mean and the variance can be calculated by the formulae

$$\mu = \frac{1}{n} \sum_{h=1}^n y_h, \quad (2.9)$$

$$\sigma^2 = \frac{1}{n-1} \sum_{h=1}^n (y_h - \mu)^2. \quad (2.10)$$

The *standard score* z_h of a data point y_h , is

$$z_h = \frac{y_h - \mu}{\sigma}. \quad (2.11)$$

Mapping a data-set with the standard score, is a standardization method, which if applied on the normal distribution, recovers the standard normal distribution. The standard score has an application in *normal probability plots*, which is a graphical tool which can be used to assess if data is normally distributed. If the plot of y_h against x_h falls on a straight line, this is an indication that the distribution can be normal. However, for a positive normality check, the points should also be symmetrically distributed on the line, and consist of a single cluster.

2.4.2 Multinomial multivariate normal distributions

The normal distribution can be generalized to variables in higher dimensions. In the 2-dimensional case, it is called the bi-variate normal distribution. For a k -dimensional variable, it is called the *multivariate normal distribution*. Given a random k -dimensional variable $\mathbf{x} = [x_1, x_2, \dots, x_i, \dots, x_k]^T$, the normal distribution takes the form

$$f(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{k/2} \sqrt{|\boldsymbol{\Sigma}|}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right], \quad (2.12)$$

where $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_i, \dots, \mu_k]^T$ is the mean vector with elements corresponding to the components of the random variable, and $\boldsymbol{\Sigma}$ is the $k \times k$ co-variance matrix with elements given by

$$\Sigma_{ij} = \text{cov}(x_i, x_j). \quad (2.13)$$

Given a set of n k -dimensional data points $[\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_h, \dots, \mathbf{y}_n]$, a numerical scheme for calculating the covariances, that is stable towards *catastrophic cancellation*, is the two pass algorithm [source], that first calculates the means

$$\mu_i = \frac{1}{n} \sum_{h=1}^n y_{h,i}, \quad (2.14)$$

which can then be used to calculate the shifted covariances

$$\Sigma_{ij} = \frac{1}{n-1} \sum_{h=1}^n (y_{h,i} - \mu_i) (y_{h,j} - \mu_j). \quad (2.15)$$

Note that the covariance matrix is symmetric about its diagonal, and that the elements along the diagonal are the variances of the individual numerical attributes. A numerical difficulty that may arise in the context of multivariate normal distributions, is when the determinant of the covariance

matrix becomes very small. The inverse of the matrix is only defined for non-zero determinants, but extremely small determinants can cause *numerical wildness* when calculating the inverse matrix. To combat this, one will typically standardize the data by subtracting the mean and scaling by the variance. Numerical wildness can still appear however, in particular when attributes are perfectly correlated¹², which is an important consideration when designing the model (perfectly correlated data is superfluous data).

For data that is multinomial as well as multivariate, each category in the range of the nominal attribute gets its own multivariate distribution. For a dataset with k numerical attributes, there will be k means and k^2 covariances, giving a total of $k + k^2$ parameters for the model. Now assume multinomial data that can take on l different values for its nominal attribute, the statistical model will now consist of $l(k + k^2)$ parameters. For large l and k values, the model can become challenging to present. Tables of numerical values are seldom useful, and hyperdimensional surfaces are hard to represent. One approach is to display the normal distributions of each numerical attribute separately, but this will not directly show the covariance behaviour.

¹²A matrix with all elements close to one, will have a determinant near zero.

3 Method

The overall development process of AutomAl 6000, spanning from its beginning as a student specialization project, to its current form as a Master’s project, has been steered mainly by the software’s capability to overlay Al-Mg-Si-(Cu) precipitates as correctly as possible, while at the same time maintaining an efficient way for the end-use researchers to analyse the validity and quality of the resulting overlays, in a manner that does not require them to check every single column¹³. There was no clear plan on how to solve the overarching problem at the outset, so to arrive at the details of the current implementation, a lot of trial and error has occurred. The eventual approach, which is outlined in this section, can be described in brevity as a symbiosis between a statistical model and a technique termed untangling, that are set together in an algorithm-friendly mathematical framework which was inspired by the digraph structure, which results in a somewhat sophisticated *column characterization* algorithm. The source code of AutomAl 6000, which embodies the implementation of the methods laid out in this section, has grown quite large in its extent, and is therefore not explained in detail here. Rather, the focus is on developing a theoretical framework of the methodology.

3.1 Column detection

The general issue of detecting atomic columns in TEM images has received much interest over the last few years. There are many benefits to effective methodologies that can perform this task, and it is an essential starting point for any automatic or semi-automatic structural characterization method. There are already several existing methods that approach this problem. Of notable mention is Atomap [12]. Atomap is sophisticated, but in simple terms, can be said to work by fitting 2D Gaussian surfaces to the pixel intensities. Despite already existing solutions, a light-weight method for column detection for AutomAl 6000 was made. This method calculates the Center Of Mass (COM) of pixel intensities around extremal pixels.

3.1.1 Image pre-processing

For the column detection to work optimally, images that are to be analyzed using AutomAl 6000 should be noise filtered using software such as Gatan Microscopy Suite (GMS) by Gatan [13], which is a commercial image processor that is widely used in the TEM-field as a constituent of the standard software suites on TEM hardware. Applying an appropriate low pass filter on the Fast Fourier Transform (FFT) of the image will eliminate many of the noise frequencies of the image. Filtering out the noise in the image is necessary for column detection to work.

To apply a low pass filter in GMS, start by performing a FFT on the image. Click on the resulting FFT with the band pass tool selected, which will produce a donut shaped mask on the FFT. Adjust the inner radius of the mask to zero, and the outer radius to approximately $6,7 \text{ nm}^{-1}$, which will

¹³If the user had to check every column in the overlay to be confident about the overlay, then that would negate the usability of our software. The point being that the users ability to understand and manipulate the underlying data effectively, has been a major factor in the development, in some cases more so than the actual accuracy of the automated parts.

include the 200 Al reflection, and exclude the 220 Al reflection. This will eliminate features that are smaller than 0.15 nm in real space. Finally, perform inverse FFT on the masked FFT to obtain the noise filtered image.

If the scale of the image is greater than 7 pm/pixel, AutomAl 6000 will automatically upsample the image so as to double both the width and height of the image. Using bilinear up-scaling has proven to have a positive effect on the column detection in images with scales in this high range. This is because the circular samples used in the COM calculations becomes over-granulated (non-circular) for low scales. AutomAl 6000 uses the resampling method of Scipy’s ndimage module [14].

3.1.2 Center of mass implementation

The input to the column detection algorithm is an image matrix M , where the elements are normalized to be floating point numbers in the range $[0, 1]$. The algorithm will also create a *deep copy* of M , termed the *search matrix* M_s , that it will use to delete already detected columns from the pixel intensity search space. The detection will continue until the maximum pixel value in M_s is below a threshold $T \in [0, 1]$.

Let the scale s of the image be given in pm/pixel, which is automatically collected from the dm3 metadata by AutomAl 6000, but can also be overwritten by the user. The *approximate atomic radii* r is then determined by

$$r = \left\lfloor \frac{100 \text{ pm/pixel}}{s} \right\rfloor, \quad (3.1)$$

where $\lfloor \cdot \rfloor$ are the floor operators that returns the integer value that leads the decimal point. The approximate atomic radii is an approximate measure of the general size (in pixels) of atomic radii in the image. This parameter determines the radius of the circular area around global maximums that will be used to calculate the COM. The *overhead parameter* o is given by

$$o = \left\lfloor \frac{7}{10} r \right\rfloor, \quad (3.2)$$

and gives the radius $r + o$ of the circular area around the found column position where the search matrix elements will be set equal to 0. The numerical values that enter into the expression for r and o are tuned to give the best results.

For the input- and parameter-space given above, the column detection method can now be described with the following recipe:

1. Create a deep copy of M as M_s .
2. Find the coordinates (x_0, y_0) of the maximum element value of the search matrix M_s .
3. Calculate the coordinates $(x_{\text{fit}}, y_{\text{fit}})$ of the centre of mass using the values from M in the subset $P_i = (x_i, y_i)$ that is formed by the circular area with radius r centered at (x_0, y_0) . If κ is the number of elements in P_i , such that $i \in [0, 1, \dots, \kappa]$, then

$$x_{\text{fit}} = \frac{1}{M} \sum_{i=0}^{\kappa} P_i x_i \quad (3.3)$$

$$y_{\text{fit}} = \frac{1}{M} \sum_{i=0}^{\kappa} P_i y_i, \quad (3.4)$$

where

$$M = \sum_{i=0}^{\kappa} P_i. \quad (3.5)$$

4. Store $(x_{\text{fit}}, y_{\text{fit}})$ as a new atomic column.
5. Set all elements in the search matrix M_s that is within a circular area of radius $r + o$ centered at $(x_{\text{fit}}, y_{\text{fit}})$, to 0.
6. Iterate the index counter.
7. Repeat step 2 to 6 until

$$\max(M_s) \leq T. \quad (3.6)$$

Some attempts at developing automatic thresholding were made, but no consistent method was found. In the current implementation, the threshold must be guessed manually. This, as well as other aspects of column detection, are discussed further in section 4.1 and 6.4.

3.2 Atomic graphs

To develop an overall algorithm that can "fit" the relational information between columns in an image to the structural principles of section 2.2.3, a framework termed atomic graph was developed. These atomic graphs are based on the digraphs of graph theory. Like digraphs, atomic graphs consist of vertices and arcs. In addition, atomic graphs have some properties that are related to the nature of this specific application. These properties are laid out here.

3.2.1 Basic atomic graph properties and vertex maps

As with digraphs, let an atomic graph be $G = (V, A)$, where $V = \{v_i\}$ is the vertex set, and $A = \{a_j\}$ is the arc set. One of the differences between an atomic graph and a general digraph, is that in atomic graphs, the vertex positions have physical interpretation. Thus, for the particular problem of this thesis, distances and angles between vertices are essential, so these values are preserved. Let the position of vertex v_i , be (x_i, y_i, z_i) relative to some arbitrary origin located within the image.

Since the vertices in an atomic graph will be representing atomic columns in projection, each vertex will have an associated relative height position in the beam direction, normal to the image plane. With one exception seen in the β' phase¹⁴, this relative z -position will either be 0 pm or $\frac{1}{2}a = 202,3$ pm, so let this displacement be represented with a Boolean value

$$\zeta_i = \begin{cases} 0 & \text{if column is in 0-plane,} \\ 1 & \text{if column is in } \frac{a}{2}\text{-plane,} \end{cases} \quad (3.7)$$

hereby referred to as the *zeta value* of the vertex.

Recall from section 2.3.2 that the distance between two vertices in a graph is an integer giving the number of vertices in the shortest path between the vertices, where "shortest" implies "fewest". In

¹⁴discussed further in section 6.5

this application, spatial separation is meaningful. Thus, let the *separation* Δ between two vertices v_k and v_l be

$$\Delta(v_k, v_l) = \sqrt{(x_l - x_k)^2 + (y_l - y_k)^2 + (z_l - z_k)^2}, \quad (3.8)$$

and let the *projected separation* Δ' between two vertices v_k and v_l be

$$\Delta'(v_k, v_l) = \sqrt{(x_l - x_k)^2 + (y_l - y_k)^2}. \quad (3.9)$$

The *symmetry number* n_i , of v_i , is the same as, and forces the out-degree of v_i , $\deg^+(v_i) = n_i$. The symmetry number of a vertex intends to reflect the number of opposite plane NN's seen in projection, as was discussed in section 2.2.3. This is a central concept in the structural principles that are being modelled with this approach, and the symmetry number of the different species under consideration, are shown in table 3.1.

Table 3.1: Atomic species and symmetry number n_i . The atomic species "Un", is a designation for Unknown, which is the initial state of every vertex.

Atomic species	n_i
Si	3
Cu	3
Al	4
Mg	5
Un	3

In addition to the definitions related to general digraphs which were discussed in section 2.3.3, like in-degree, out-degree, in-neighbourhood, etc, some additional set definitions now follow. These facilitate the mapping of neighbourhoods, which inform some of the methods explored in section 3.4. Consider first a *district* D , which is an ordered list of vertices that are local to a particular vertex at a level above neighbourhoods, with elements D_k . The term "local" is used here in a purposefully vague sense. The column characterization method progresses by making changes to the district, so one could consider this list to contain local vertices sorted by their currently attributed relevance. At the outset however, the district D of v_i shall consist of the 10 "closest" vertices to v_i , where "closeness" is understood to be measured in projected separation Δ' . The out-neighbourhood N^+ of v_i shall then be defined as the set containing the first n_i elements of $D(v_i)$. By using set-builder notation, this can be written as

$$N^+(v_i) = \{k \in [1, 2, \dots, n_i] | D(v_i)_k\}. \quad (3.10)$$

The in-neighbourhood N^- of v_i is then defined by

$$N^-(v_i) = \{v_j \in (V(G) - v_i) | v_i \in N^+(v_j)\}. \quad (3.11)$$

Let the neighbourhood N of v_i be the union of the in- and out-neighbourhood,

$$N(v_i) = N^+(v_i) \cup N^-(v_i). \quad (3.12)$$

If a vertex v_l is in the intersection of the in- and out-neighbourhood of a vertex v_k , they are *partners*. The set of all partners P of v_i , is then

$$P(v_i) = N^+(v_i) \cap N^-(v_i). \quad (3.13)$$

Furthermore, let the *anti-neighbourhood* $\overline{N}(v_i)$ of v_i be

$$\overline{N}(v_i) = \{v_j \in D(v_i) | v_j \notin N(v_i)\}, \quad (3.14)$$

and let the *semi-partners* \tilde{P} of v_i be

$$\tilde{P}(v_i) = N(v_i) - P(v_i). \quad (3.15)$$

Two final sets that are meaningful here, is the *out-semi-partners*

$$\tilde{P}^+(v_i) = \tilde{P}(v_i) \cap N^+(v_i) \quad (3.16)$$

and the *in-semi-partners*

$$\tilde{P}^-(v_i) = \tilde{P}(v_i) \cap N^-(v_i). \quad (3.17)$$

The sets that have been defined here are summarized in table 3.2. This turns out to be a convenient way to build the implementation of atomic graphs. Note that the district is an ordered list (not a set) from which the sets in the map of v_i are defined.

Table 3.2: Sets that together map out the local graph relationships for a given vertex v_i

Set	Definition	
District	$D(v_i) = [\dots]$	
Out-neighbourhood	$N^+(v_i) = \{k \in [1, 2, \dots, n_i] D(v_i)_k\}$	(3.10)
In-neighbourhood	$N^-(v_i) = \{v_j \in (V(G) - v_i) v_i \in N^+(v_j)\}$	(3.11)
Neighbourhood	$N(v_i) = N^+(v_i) \cup N^-(v_i)$	(3.12)
Anti-Neighbourhood	$\overline{N}(v_i) = \{v_j \in D(v_i) v_j \notin N(v_i)\}$	(3.14)
Partners	$P(v_i) = N^+(v_i) \cap N^-(v_i)$	(3.13)
Semi-partners	$\tilde{P}(v_i) = N(v_i) - P(v_i)$	(3.15)
Out-semi-partners	$\tilde{P}^+(v_i) = \tilde{P}(v_i) \cap N^+(v_i)$	(3.16)
In-semi-partners	$\tilde{P}^-(v_i) = \tilde{P}(v_i) \cap N^-(v_i)$	(3.17)

As can be surmised from table 3.2, once the districts and symmetry numbers are defined, the entire atomic graph is defined. However, there is more than one conceivable way to select the out-neighbourhood, generating a whole family of graphs, with one useful in particular, discussed in section 4.3.2.

At last, introduce a restriction on atomic graphs, which in fact is a very severe restriction. Arcs in atomic graphs can not intersect. One of the shortcomings of the definition of the map of v_i , is that it does not protect against intersecting arcs. Checking for arc intersections is a computationally difficult task that the current implementation solves by geometrically calculating the intersection between

every arc and then checking the domain of the intersection. Discovering a method, for instance by using the adjacency matrix of the atomic graph that would simplify intersection detection, would contribute a major speed increase of the AutomAl 6000 column characterization. This restriction is a result of the application, and the interpretation of an arc as indicating a nearest neighbour atom pair in an atomic structure.

Atomic graphs shall be visualized by circles representing the vertices, that are filled or hollow, with filled circles indicating $\zeta_i = 0$, and hollow circles indicating $\zeta_i = 1$. Symmetrically adjacent vertices, meaning that they are both adjacent to each other, shall be connected by a simple black line. Thus, single black lines indicate two arcs, one arc and its inverse. It is desirable that un-symmetric arcs stand out, so these are given a red color, bold face and an arrowhead indicating the "adjacent to" direction. Arcs connecting vertices that occupy the same plane are indicated by a blue bold face and arrowheads, but un-symmetric arcs take precedence, so only symmetric in-plane arcs will ever appear in blue. Some examples of an atomic graph visualisation is shown in figure 3.1.

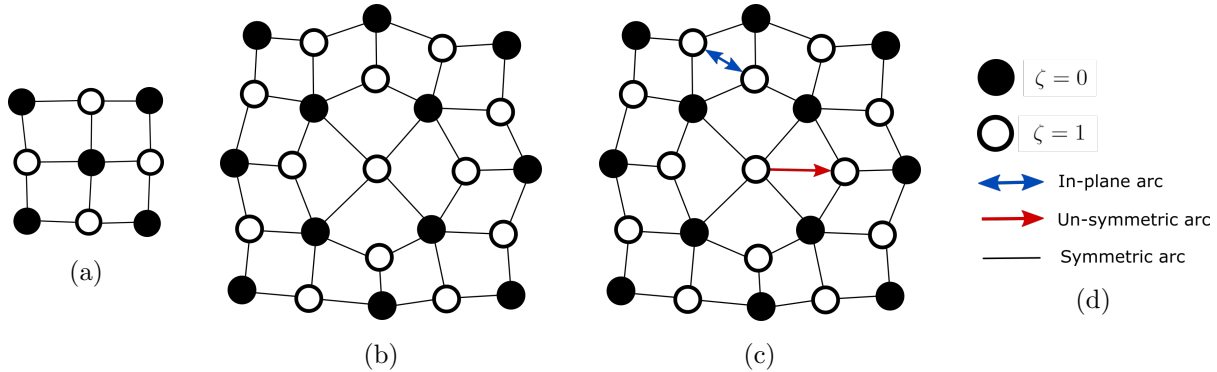


Fig. 3.1: Some examples of atomic graph visualizations. Note that the edge symmetries are wrong. This is because edge effects are not considered at all, and the graph representation is assumed to extend indefinitely. (a) Atomic graph representation of a FFC unit cell. Similar to figure 2.4a. (b) A β'' eye, similar to figure 2.4d. Note that when atomic graphs represent structures that are consistent with the principles of section 2.2.3, they are symmetric, ie no un-symmetric arcs. (c) The same β'' eye, but with some erroneous arcs. From this graph one can see that the central column has $n_i = 5$, meaning that its species is set to Mg, which is incorrect for this Al position. (d) Legend.

3.2.2 Subgraph classifications

Subgraphs can be produced from a graph by different methods, see section 2.3.2. Of special interest here, is the induced sub-graph method, which from a given vertex subset, $V(H) \subseteq V(G)$, will produce an arc set $A(H)$ given by equation 2.5. The induced subgraph H is then given by $H = (V(H), A(H))$. The following paragraphs explain how to find certain vertex subsets, which then is assumed to imply the induced subgraphs from those vertex subsets. There are three categories of vertex subset definitions of special interest here.

Mesh centered subgraph

Given the no-arc-intersection restriction on atomic graphs and the spatial property of vertices, a mesh

can be defined rather easily in atomic graphs as an area enclosed by arcs. Finding the vertex subset of a mesh is done by first defining an arc $v_l v_k$, and then selecting the vertex v_q from $N(v_k)$ which minimizes the vector angle between the vectors formed by $v_k \rightarrow v_l$ and $v_k \rightarrow v_q$. This is repeated until $q = l$. Think of this as going from v_l to v_k and then taking a rightmost turn along any arc until v_l is reached. The induced subgraph formed by such a vertex subset, is a 1st order *mesh centered subgraph*. A 2nd order mesh centered subgraph would include all adjacent meshes, but only 1st order subgraphs are used by AutomAl 6000, so these are not covered here. Figure 3.2 provides some examples of the procedure. For notation, let the 1st order mesh centered subgraph defined from the vertex pair (v_i, v_j) be $H_{\text{mesh}}^{(1)}(v_i, v_j)$, where v_i and v_j is assumed to be adjacent in at least one way.

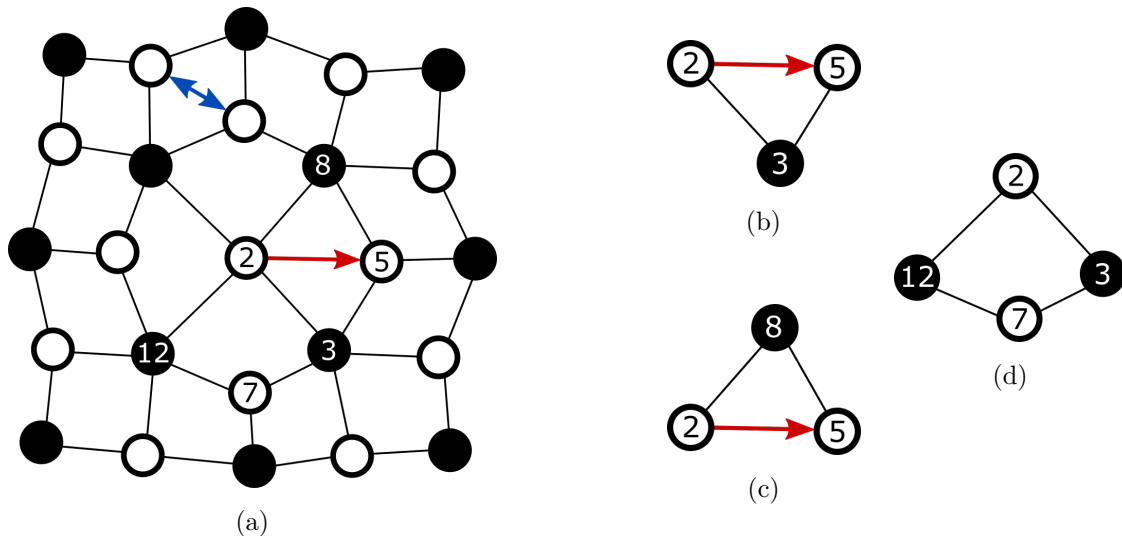


Fig. 3.2: Examples of 1st order mesh centered subgraphs. (a) The same atomic graph G , as in figure 3.1c, with some of the vertices labelled with arbitrary indices. (b) The 1st order induced mesh centered subgraph $H_{\text{mesh}}^{(1)}(v_5, v_3)$. Note that $H_{\text{mesh}}^{(1)}(v_5, v_3) \neq H_{\text{mesh}}^{(1)}(v_3, v_5)$. (c) The 1st order mesh centered subgraph $H_{\text{mesh}}^{(1)}(v_5, v_2)$. Note that this subgraph can be defined by the vertices $v_5 v_2$, even though $v_5 v_2$ is not an arc in $A(G)$, however, its inverse is, which is the minimum requirement. As a counterexample, the subgraph $H_{\text{mesh}}^{(1)}(v_8, v_3)$ does not exist. (d) The 1st order mesh centered subgraph $H_{\text{mesh}}^{(1)}(v_{12}, v_2)$. Note that $H_{\text{mesh}}^{(1)}(v_{12}, v_2) = H_{\text{mesh}}^{(1)}(v_2, v_3) = H_{\text{mesh}}^{(1)}(v_3, v_7) = H_{\text{mesh}}^{(1)}(v_7, v_{12})$.

Arc centered subgraph

A 1st order *arc centered subgraph*, is formed by the vertex subset that includes an arc $v_l v_k$, as well as the meshes formed by $v_l v_k$ and $v_k v_l$. This is illustrated in figure 3.3. The 2nd ordered arc centered subgraph additionally includes all the vertices of the 1st order arc centered subgraphs formed on each arc except the original defining arc. This is illustrated in figure 3.3c. Let the notation for arc centered subgraphs be $H_{\text{arc}}^{(\text{order})}(v_l, v_k)$.

Vertex-centered sub-graph

A 1st order *vertex centered subgraph* $H_{\text{vertex}}^{(1)}$, is formed by a vertex v_i , as well as all arc centered subgraphs formed by $v_i v_j$ for all $v_j \in N(v_i)$. This is illustrated in figure 3.4.

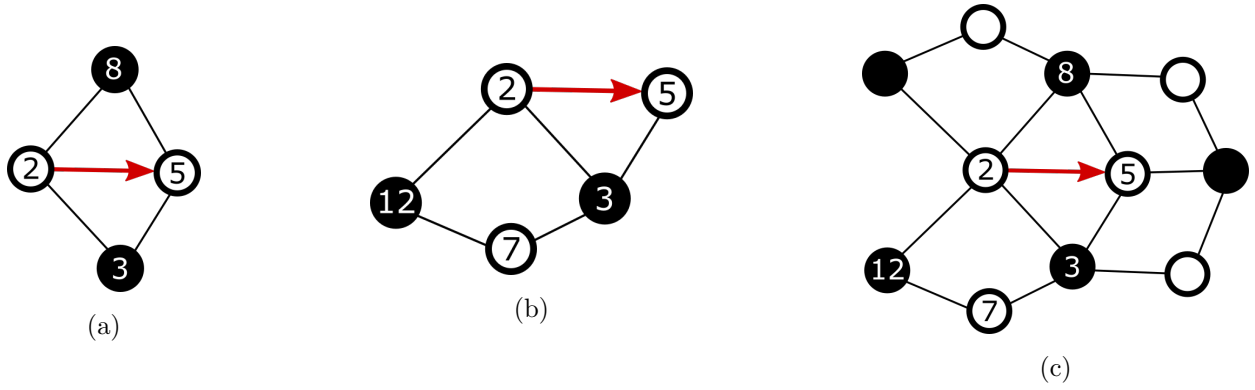


Fig. 3.3: Examples of arc centered subgraphs induced from G in figure 3.2a. (a) The 1st order induced arc centered subgraph $H_{\text{arc}}^{(1)}(v_2, v_5)$. Note that in contrast to mesh centered subgraphs, $H_{\text{arc}}^{(1)}(v_2, v_5) = H_{\text{arc}}^{(1)}(v_5, v_2)$. (b) The 1st order induced arc centered subgraph $H_{\text{arc}}^{(1)}(v_3, v_2)$. (c) The 2nd order induced arc centered subgraph $H_{\text{arc}}^{(2)}(v_2, v_5)$.

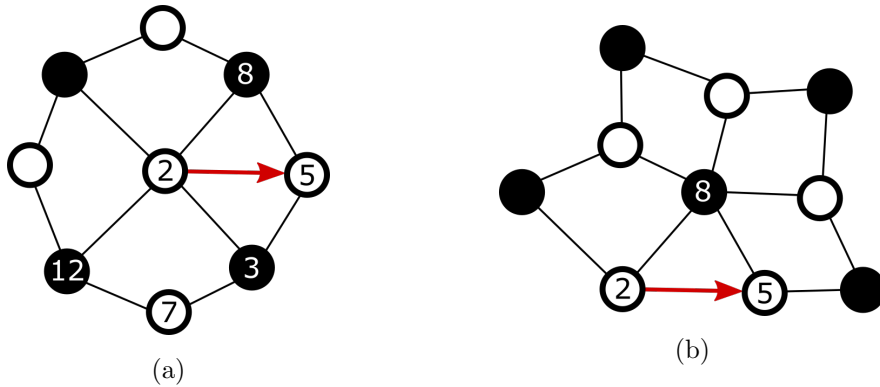


Fig. 3.4: Examples of vertex centered subgraphs induced from G in figure 3.2a. (a) The 1st order induced vertex centered subgraph $H_{\text{vertex}}^{(1)}(v_2)$. (b) The 1st order induced vertex centered subgraph $H_{\text{vertex}}^{(1)}(v_8)$.

3.2.3 Numerical vertex attributes

The vertices of atomic graphs have several associated numerical attributes which can be assumed to be correlated to the symmetry number or atomic species. The angles formed in the graph, as well as the pixel intensity in the same image location, is among these, and they are defined below.

Alpha angles

As was discussed when introducing atomic graphs, the symmetry number n_i determines the out-degree of the vertices and places this number in the range $\{3, 4, 5\}$. This guarantees that every vertex v_i will be adjacent to at least three other vertices, forming at least three arcs $\{a_j\} = v_i\{v_k\}$, where $v_k \in N^+(v_i)$. The three angles in between these arcs can be calculated, and the minimum and maximum of these angles correlate to the symmetry number n_i , which again is related to the species of the column. Let these 3 angles be the *alpha angles*, and they are illustrated in figure 3.5. The distribution of this attribute is explored further in section 3.3.

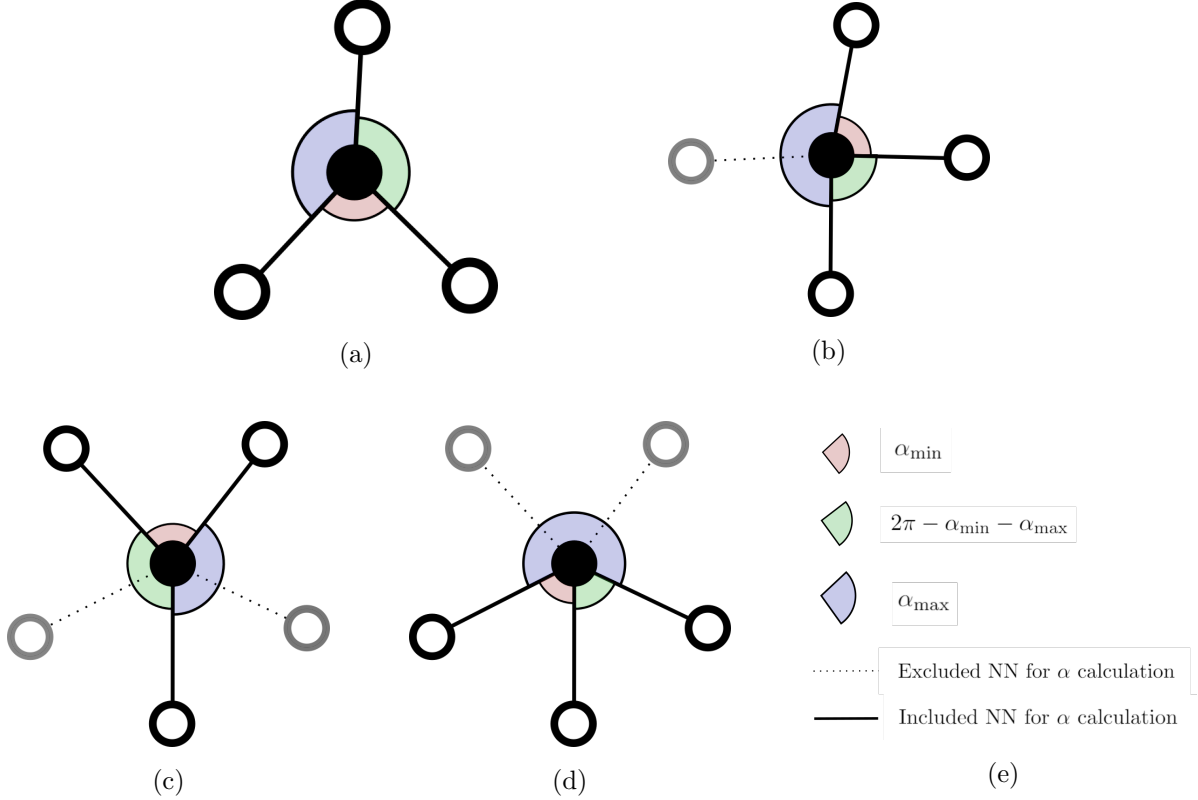


Fig. 3.5: Alpha angles are defined as the angles between the vectors formed by a vertex and the three first vertices in its district. Assuming that these vertices are opposite plane NN's, the symmetry number of the vertex will effect the expectancy value of these angles. (a) A vertex with symmetry number $n_i = 3$. The expectancy of both α_{\min} and α_{\max} should be around $\frac{2\pi}{3}$ radians. (b) A vertex with symmetry number $n_i = 4$. Here the expectancy of α_{\min} should be around $\frac{\pi}{2}$ radians, while the expectancy of α_{\max} should be around π radians. (c) For vertices with symmetry number $n_i = 5$, there are two scenarios that can occur. In both scenarios the expectancy of α_{\min} should be around $\frac{2\pi}{5}$ radians. In this scenario, the expectancy of α_{\max} should be around $\frac{4\pi}{5}$ radians, while in the other scenario, (d), the expectancy of α_{\max} should be around $\frac{6\pi}{5}$ radians. (e) Legend.

Theta angles

In the column-centered sub-graph of a vertex v_i , the central angles formed between the arcs extending into or out from v_i , should once again provide angles that should give clear indications of n_i . Of particular interest is the minimum, maximum and average of these angles, here termed the *theta angles*, with special interest in the maximum θ_{\max} , minimum θ_{\min} and the average $\bar{\theta}$. In contrast to the alpha angles, which are used to get a statistical prediction from nothing more than projected distances, the theta angles are a bit more selective. Only angles that span a mesh corner that is consistent with the structural principles¹⁵, meaning that they have a vertex subset cardinality of 4. Considering again figure 3.4a, this would exclude the angles $\angle(v_3, v_2, v_5)$ and $\angle(v_5, v_2, v_8)$, successfully eliminating the erroneous data from being used in the classification. By the same token, it would also eliminate the non-erroneous angle $\angle(v_2, v_8, v_5)$ from figure 3.4b. The distribution of the theta angles is explored in more detail in section 4.3.1.

¹⁵See item 5 in section 3.2.4.

Gamma

So far, the θ and α angles give some indication of the symmetry number n_i , which then can be related back to atomic species. Knowing the symmetry number does not guarantee knowledge about the species though. For instance, since both Cu and Si columns have $n_i = 3$ in the structural model, another attribute is needed to separate these. Luckily the column intensity in HAADF-STEM images, are related to the atomic number of the atomic species that occupies the column. Si and Cu differ a lot in atomic numbers compared to the other alloying elements of the systems under consideration here, meaning that the intensity can hopefully be used to assess species. Before one can use intensities as an advantage in this characterization though, there are some hurdles to overcome.

The value of the brightest pixel in a column, is termed the peak gamma. The average pixel values within the circle with radius r centered at the column position, is here termed the average gamma, with r being the same approximate atomic radii as discussed in section 3.1.2. In an attempt to make the gamma values directly useful in the statistical model, define a gamma normalization standard. Seeing as how the intensities are relative to the baseline of each image, as well as the sample thickness etc, some simple assumptions are made, that despite lack of merit, turns out to be useful. Measure first the intensities, excluding columns in the particle, and calculate a normalization factor such that the average intensity of the matrix takes on a pre-determined value. This obtains the *normalized peak gamma* γ_{peak} and the *normalized average gamma* γ_{avg} . Considering some of the factors that this procedure ignores, like Cu-rich columns in the matrix potentially shifting the normalization, or sample with different relative thickness, the resulting distributions will have a higher variance, but the method still seems useful. This is discussed further in section 3.3.

3.2.4 Structural principles in atomic graphs

When representing an atomic structure with an atomic graph, it is with each NN in the opposite plane included in the partner set of the corresponding vertex. Following this logic, the entire graph should include the inverse of each arc, and therefore be symmetric. The consequences of the structural principles, as discussed in section 2.2.3, in terms of their atomic graph representation, can then be formulated in the following manner.

1. For every vertex $\{v_i\}$, $\text{deg}^+(v_i) \in \{3, 4, 5\}$.
2. For every arc $\{a_j = v_k v_l\}$, the component vertices must belong to opposite planes such that $\zeta_k \neq \zeta_l$.
3. For every arc $\{a_j = v_k v_l\}$, the arc set must contain its inverse, ie $v_l v_k \in A(G)$, which implies that G is symmetric.
4. Arcs can not intersect in their geometrical representation, given meaningful vertex positions in a 2D plane.
5. The order of any mesh centered subgraph must be $|V(H_{\text{mesh}}^{(1)})| = 4$.

3.2.5 Graph operations

Section 3.4.2 attempts to develop a column characterization algorithm on the atomic graph framework, which requires the definition of some elementary operations. Let these graph operations be categorized as *weak operations* if they are limited to changing the successor of arcs, and as *strong operations* if they are enabled to also change symmetry numbers.

Arc permutation

The methods of AutomAl 6000 sometimes calls for an *arc permutation*, which means to change which vertex an arc is adjacent to, without changing the *pivot vertex* which the arc is adjacent from. In the vertex map implementation of atomic graphs, this involves permuting two elements, the original successor and the new successor, in the district of the pivot vertex, and then updating all sets in the area to reflect the change and to ensure that the definitions of table 3.2 remains true. Let the notation for an atomic graph arc permutation of the arc be $\hat{P}_i(j, k)$, where i is the index of the pivot vertex, j is the index of the original successor vertex of the arc, while k is the index of the new successor vertex. An illustration is given in figure 3.6.

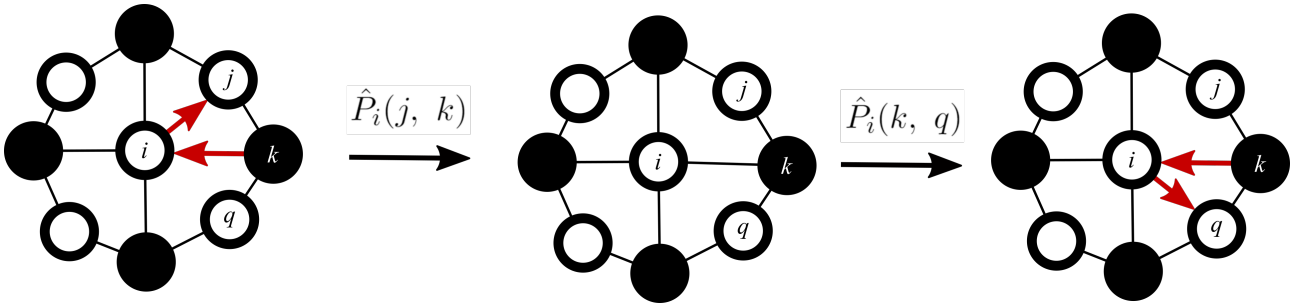


Fig. 3.6: Illustration of the effect of arc permutation operations on an atomic graph. Note that one can not permute an arc into an arc that already exists, nor permute an arc that does not exist.

Weak arc termination

Weak arc termination is concerned with removing a specific arc, without changing any symmetries, which in effect means to relocate the arc, rather than to terminate it, despite this methods designation. To do this, AutomAl 6000 uses a set of rules in an attempt to find an alternative arc direct successor. If no alternative is found, the arc remains unaltered. Let $v_i v_j$ be an arc subject to weak arc termination. Furthermore, let the search-space S for an alternative successor v_k , be the set

$$S = V(H_{\text{vertex}}^{(1)}(v_i)) - N^+[v_i], \quad (3.18)$$

where $H_{\text{vertex}}^{(1)}(v_i)$ is the 1st order vertex centered subgraph of v_i , and $N^+[v_i]$ is the closed out-neighbourhood of v_i . If there are elements in the intersection of the in-neighbourhood of v_i and S , then these are prioritized as long as the "would be" arc $v_i v_k$ will generate the desired cardinality in the subsets $V(H_{\text{mesh}}^{(1)}(v_i v_k))$ and $V(H_{\text{mesh}}^{(1)}(v_k v_i))$, which should be 4. If a suitable candidate within parameters is found, perform the arc permutation $\hat{P}_i(j, k)$.

Weak arc preservation

Weak arc preservation on a un-symmetric arc $v_i v_j$, will attempt to find an arc $v_j v_q$ that is suitable for permutation $\hat{P}_j(q, i)$, which would create the inverse arc of $v_i v_j$, namely $v_j v_i$, thus making it symmetric. The search space S for v_q will be

$$S = \tilde{P}^+(v_j), \quad (3.19)$$

where $\tilde{P}^+(v_j)$ is the out-semi-partners of v_j . This set does not include v_j , because $v_i v_j$ is assumed to be un-symmetric for this operation to be meaningful. If a vertex v_q in S is found to be such that both $H^{(1)\text{arc}}(v_j, v_q)$ and $H_{\text{arc}}^{(1)}(v_q, v_j)$ have cardinality 4, then the arc permutation $\hat{P}_j(q, i)$ is performed.

Strong arc termination

Strong arc termination attempts to remove an arc $v_i v_j$, by reducing the symmetry number of v_i , and thus also the associated atomic species, which is the defining feature of a strong operation. If this cannot be done, for instance if n_i is already equal to the minimum number in the range of the symmetry number, then no action is performed.

Strong arc preservation

Strong arc preservation on an arc $v_i v_j$, attempts to preserve the arc by increasing the symmetry number n_j of v_j . If this is not possible, no action is performed.

3.3 Statistical model for numerical vertex attributes

AutomAl 6000 features a flexible data management module, which can be used to construct custom statistical models from selected data. In this section, the specifics of the *default model* which is featured with AutomAl 6000, are discussed.

Being asked to produce an atomic overlay, is in essence a classification problem. For a given data point, the statistical model must produce a probability vector with probabilities of how likely it is that the data point belongs to a certain category of the nominal attribute. It might seem obvious that the nominal classification attribute is the atomic species of the column. However, for both physical reasons, and practical reasons¹⁶, the nominal attribute which the default model classifies by, is arbitrarily termed the *advanced species*, and is a higher resolution classification system that aims to capture details that would break normality in a less considerate classification system. For instance, the distribution of Si α -angles in Q' , is better modelled with two normal distributions, than with one, due to the difference between the structural functions between the in-plane NN Si-Cu position, and the Si without Cu as NN. This is because the structural environment is different between these positions, thus systematically causing some "stretching" in the triangular symmetry, making it less symmetric in the case of in-plane Si-Cu, which in turn effects the expectancy values of the α -angles. Another example is the α_{max} attribute for Mg, which can randomly take on one of two different expectation

¹⁶The way in which the numerical attributes are defined can produce distributions with multiple peaks (not simply normal).

values, due to how α -angles are defined¹⁷. In this scheme then, each advanced species category must map to an atomic species, and a symmetry number. The categorization used in the default model is shown in table 3.3. This particular scheme is supported by experience and extensive trial and error.

Table 3.3: The range and mapping of the advanced species nominal attribute used by the default model in AutomAl 6000. This table is the *species dictionary* of the default model. Custom user models may have a different species dictionary.

Advanced species	Atomic species	n_i	description
Si ₁	Si	3	Normal Si as found in β'' and interfaces
Si ₂	Si	3	In-plane NN to Cu in Q' and C phases
Cu ₁	Cu	3	
Al ₁	Al	4	Al in FCC matrix positions
Al ₂	Al	4	Al inside the precipitate
Mg ₁	Mg	5	Mg with $\alpha_{\max} \leq 3.12$
Mg ₂	Mg	5	Mg with $\alpha_{\max} > 3.12$
Un ₁	Un	3	"Unknown" initial state

Table 3.4: The numerical vertex attributes used by the default model in AutomAl 6000. (*Not used in the default model, but listed here because its an available attribute).

Numerical attribute	Symbol
Alpha max	α_{\max}
Alpha min	α_{\min}
Theta max*	θ_{\max}
Theta min*	θ_{\min}
Average theta	$\bar{\theta}$
Normalized peak gamma	γ_{peak}
Normalized average gamma	γ_{avg}

Consider now some actual data from a sample of HAADF-STEM images which are presented in figure 3.7. This is a very small sample, but serves to illustrate some of the important considerations when modelling. This small sample includes Q' , β'' , L , β'_{Cu} and some general disorder. The numerical attributes considered in the default model are summarized in table 3.4. This particular choice of numerical attributes is also highly deliberate with regards to avoiding numerical wildness in the covariance matrices, as was discussed in section 2.4.2. From the overlays of the images in figure 3.7, where both the overlays and atomic graphs have been manually corrected, and for each advanced species category, a mean vector μ and a covariance matrix Σ is calculated using equation 2.14 and 2.15. The determinant $|\Sigma|$ and inverse covariance matrix Σ^{-1} is then calculated with Scipy's linalg module [14].

¹⁷This classification consideration is illustrated with more clarity in section 4.3.1, where the normality assumption is also challenged.

In figure 3.8, the results of this are shown with the individual attributes fitted to 1D normal distribution curves. This model can now make predictions on new sets of numerical attributes by using equation 2.12 with the appropriate parameters for each advanced species, thus producing a probability vector. The model can also make predictions from incomplete attribute data. By substituting missing data with the appropriate means, the model gives the best possible prediction from the data. This is especially useful at the initial stages of the column characterization, when only α -angles are available. More on this in section 3.4.3.

3.4 Column characterization

Structural principles and vertex statistics, both now set in atomic graphs separately, are in this section joined together to build an algorithm that can produce a column characterization¹⁸ of a HAADF-STEM image. The column characterization consists of a whole slew of sub-algorithms. These are not all explained in their methodical detail, rather, their purpose is explored. The main feature of the algorithm, termed untangling, are explained in more detail however, as well as the eventual arrangement of methods.

3.4.1 Sub-methods

Spatial mapping

The spatial mapping assigns the initial district of each vertex in the atomic graph. As was discussed in section 3.2, this involves finding the 10 closest vertices of each vertex, where closeness is measured in projected separation. This implementation calculates the projected separation between all the vertices of the graph and stores the values in the *projected separation matrix*, which is a $n \times n$ matrix with elements

$$\delta'_{ij} = \delta'_{ji} = \begin{cases} 0 & \text{if } i = j, \\ \Delta'(v_i, v_j) & \text{if } i \neq j, \end{cases} \quad (3.20)$$

where n is the order of the atomic graph. The initial district of vertex v_i will then be the vertices with column indices corresponding to the 10 smallest values along the row i .

Precipitate detection

It is important for many aspects of the analysis, that the particle can be identified in distinction from the aluminium matrix. Of most notable importance, is the need to assess exclusively the matrix pixel intensities so as to determine the normalized gamma attributes. Any vertex labelled as Al as its atomic species, is labelled as belonging to the precipitate if it has more than one non-Al vertex in its neighbour set.

¹⁸”Column characterization” here indicate a characterization of a precipitate by determining column properties (Species and 3-d position).

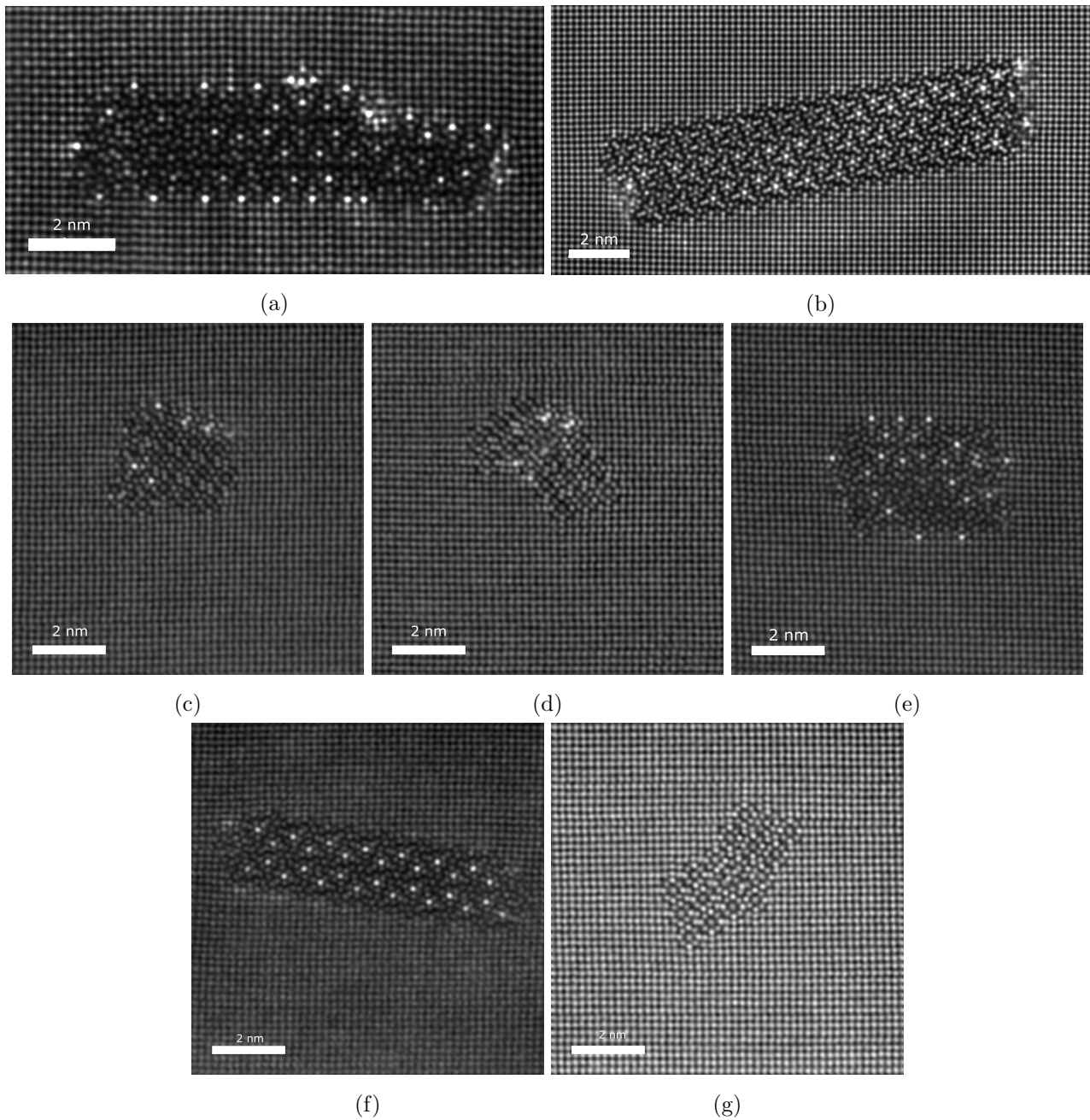


Fig. 3.7: A small data sample. (a) A large L -phase. (b) A highly ordered and large Q' precipitate, attenuated with *smart align*, which is an image acquisition technique which aligns information from several images. (c) A hybrid particle with two β'_{Cu} stars at the top, a Q' Cu position, some β'' and some disorder. (d) A β'' particle with some β'_{Cu} . (e) A hybrid precipitate with L and Q' . (f) A medium Q' . (g) A β'' .

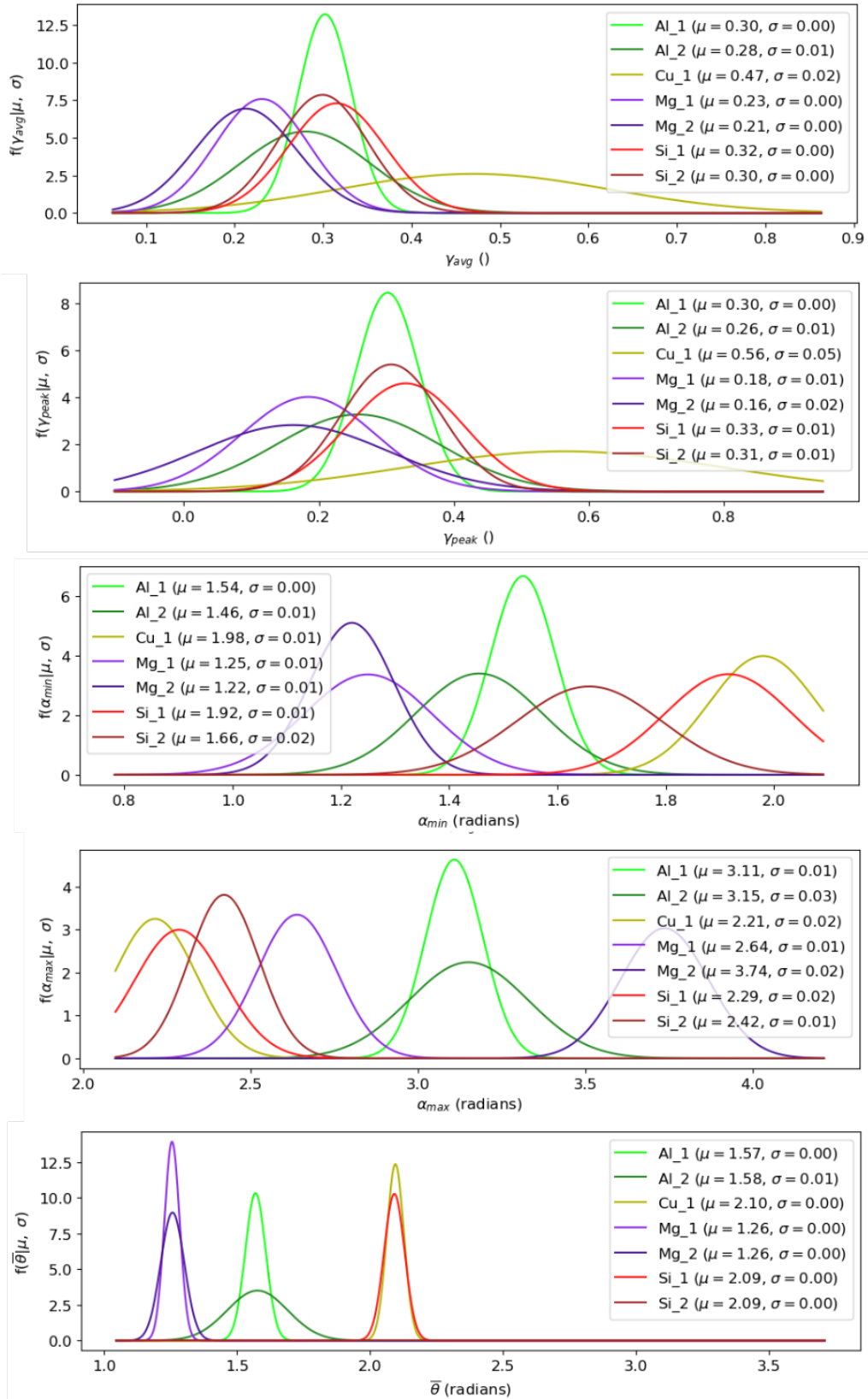


Fig. 3.8: Individual attribute projection of the default model multinomial multivariate normal distributions. Note the separation of the means between Mg₁ and Mg₂ in the α_{max} attribute, as well as the separation between the Si₁ and Si₂ means in the α_{min} attribute. The variance of the $\bar{\theta}$ attribute is manually inflated, because of course, the average theta angle in a symmetric graph, will always be 2π .

Zeta analysis

Determining the Boolean ζ -value of each vertex is an integral part of the problem, that is, determining which plane a column belongs to, see section 3.2. The idea behind this sub-method, is a repeated election system. Start with a specific vertex, and give this vertex a vote of 1. All other vertices then have a vote of 0. Then let every vertex give a vote to the vertices in its partner set. If a vertex has 0 votes, its partners will also receive 0 votes. Thus, on the first election, only the partners of the single vertex with vote 1, will receive votes. On the next round, the original vertex will receive additional votes from its partners, who now have voting power, strengthening the voting power of the original vertex in a positive feedback loop. This causes the original vertex to act as a source for the entire graph, and thus making sure that the vertex zeta states are relative to the original vertex, and not only to their immediate local vertices. Repeat the elections several hundred times to achieve a convergent voting state across the graph. This method has been termed *zeta analysis*. This discussion is limited to this short qualitative explanation, because it is at the very forefront of the most recently conceived methods of AutomAl 6000, and as such is not yet fully refined, as is necessary for a technical discussion. The zeta analysis is very promising though, and performs very well, until it does not. If there is a point in the graph that changes the polarity of the voting, some areas of the graph will have inverted zeta values. This means that the more accurate the graph is, the more robust the zeta analysis is. There is perhaps some avenues of improvements to this method, which could potentially make it effective and reliable at determining the column planes, even when only projected separations are known. This could lead to a more general column characterization algorithm, and the possibility of this is discussed in section 4.3.2.

Arc intersection detection and termination

For each arc adjacent to or from each vertex, this method calculates the intersection point between the arc and the arcs adjacent to or from the neighbours of the vertex, as well as the neighbours of the neighbours of the vertex. If an intersection is found, this intersection is eliminated by somewhat brutally changing the atomic species of the involved vertices.

Sub-set mapping

The map of each vertex needs to be regularly updated as changes to the graph occur. This update occurs often during column characterization, and it simply uses the definitions of table 3.2 to assign these sets throughout the graph.

3.4.2 Untangling

Given an atomic graph G , the goal is to alter it in such a way that all the atomic graph rules given in section 3.2.4 are obeyed. An atomic graph that correctly describes an atomic structure, should be symmetric and therefore only contain symmetric arcs. To summarize the method, which is explained in more detail below, then for every un-symmetric arc $v_i v_j$ in G , analyse the 1st order arc centered subgraph. These subgraphs can be categorized according to the order of the meshes in the subgraph, and again sub-categorized according to the content of symmetric and un-symmetric arcs in the meshes. This provides AutomAl 6000 with a map of which operation to perform for each given subgraph. This

has been termed *untangling*. If AutomAl 6000 encounters a subgraph that is not within the map, no operation is performed.

Finding all the un-symmetric arcs in a graph is very straight forward, given the vertex maps. Using set building notation, all un-symmetric arcs $v_i v_j$ of G , can be found in the following manner

$$\{v_i v_j\} = \{v_i v_k \in A(G) \mid v_k \in \tilde{P}^+(v_i)\}. \quad (3.21)$$

For each of these arcs, produce the 1st order arc centered subgraph $H_{\text{arc}}^{(1)}(v_i, v_j)$. Now, classify the subgraph according the combination of the cardinality of the meshes, and call this its *class*. Furthermore, classify the specific configurations of symmetric and un-symmetric arcs in the subgraph, and call this the *configuration* of the subgraph. The current version uses the classes and configurations that are shown in figure 3.9, in order to determine how to attempt to resolve the original un-symmetric arc $v_i v_j$. The classes are given arbitrary numbers, while the configurations are given arbitrary capital letters within each class. Each vertex in the subgraph is assigned lowercase letters in each class, except i and j , which are reserved for the subgraph-defining arc. The classes and configurations are not labelled with their alphanumerical designations in the figure, since these are arbitrary. Figure 3.9 is meant to illustrate the general idea with untangling, rather than to document its actual content. As can be surmised, only a minority of the configurations have an associated strong operation. The idea is that weak untangling should bring the most common graph configurations into more predictable ones, and that strong untangling will only be applied to highly predictable configurations. The strong operations are suppressed as much as possible, because it will compete and sometimes override the predictions of the statistical model. Weak untangling is always performed first, preferably more than once, before strong untangling is done.

There is no rigid logic behind the specific contents of the untangling map, rather, it is arrived at through consideration of actual results. There is a balance here, on the one hand, one could solve any specific subgraph by adding the appropriate operation to the map, while on the other hand, this could potentially break generality and produce unpredictable results in other cases. The configurations that are added in the map shown here, all have mostly predictable behaviours. This does not amend the fact that this untangling approach is imperfect. One strategy is to extend the range of the map, perhaps even into 2nd order subgraphs, but this has a taste of unrestrained growth in complexity. Another strategy is to circumvent untangling with other methods, which is discussed in section 4.3.2.

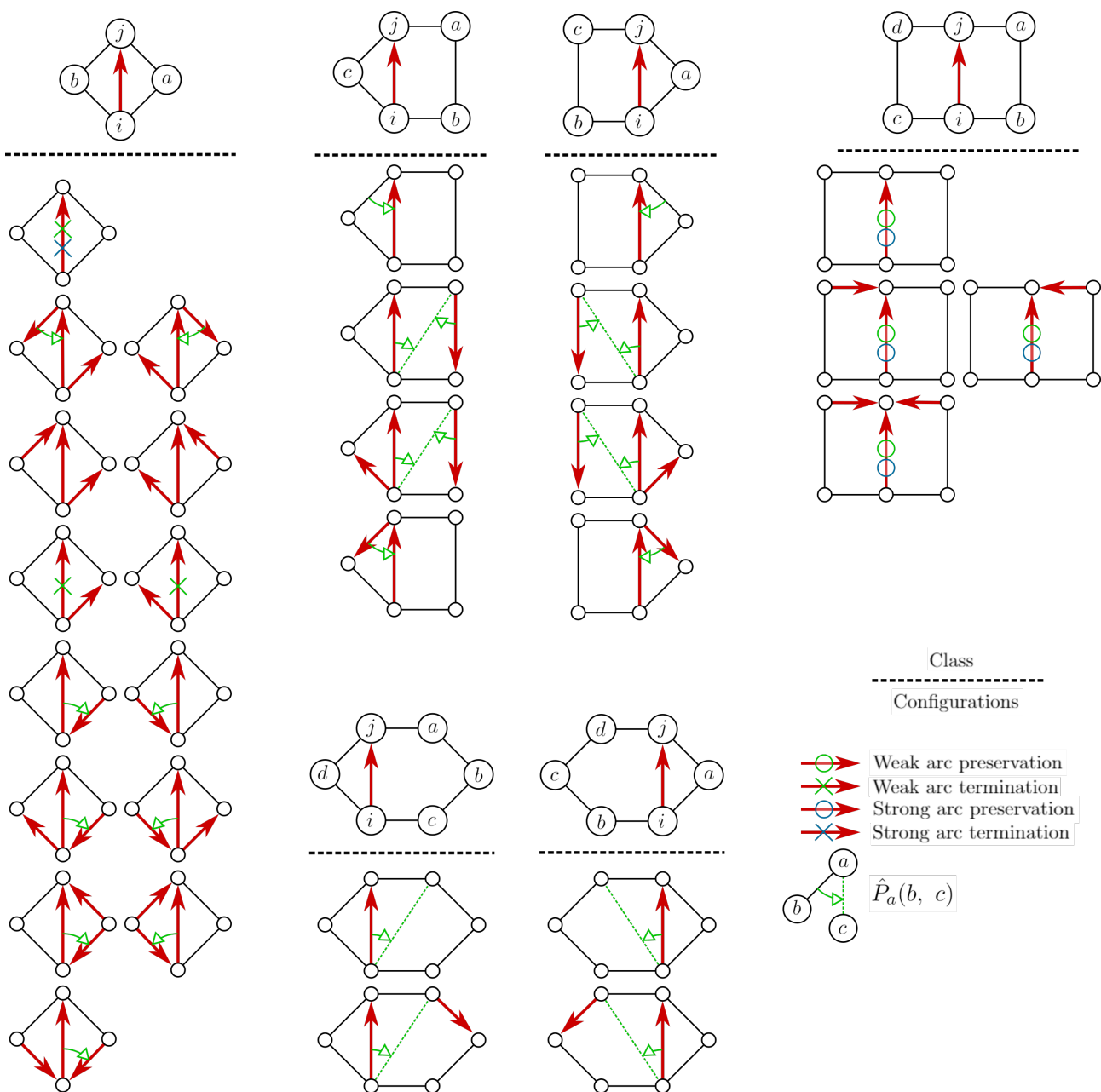


Fig. 3.9: Map of the arc centered subgraph classifications by class and configuration, which is used by AutomAI 6000 for untangling atomic graphs. Neither the classes nor the configurations are exhaustive, but capture the most common subgraph configurations encountered in un-symmetric atomic graphs in a practical sense. The legend in the lower right corner explains which operation AutomAI 6000 will attempt to perform for each given configuration during weak or strong untangling.

3.4.3 Composite algorithm

The column characterization algorithm of AutomAl 6000 is a combination of all the methods that have been discussed previously. The sequence in which these methods are run, encrypts a complex behaviour, and different sequences will produce different results. The general idea is to take a very general starting point, with every vertex defined with the atomic species Unknown (Un), and thus symmetry numbers set to 3. Use the alpha angles to make an initial classification prediction with the default model across the graph. This classification is only expected to be as good as the graph information, so will produce many erroneous predictions. Next, apply untangling and thus hopefully improve the symmetry of the atomic graph, so that the graph will better represent the actual atomic structure. Now, the model is supplied with more reliable numerical vertex attributes, which improve the prediction capability of the model. Untangling is then applied again, and so on, until the method converges at its best result. However, this is only the qualitative thinking behind the particular sequence below, which of course has a much more complex underlying dynamic to it, which is highly nontrivial to unravel in a technical sense. The currently implemented sequence for the full column characterization algorithm, which assumes an accurate column detection, is:

1. **Spatial mapping.** Calculate the projected separation matrix of the graph, and assign the initial vertex districts.
2. **Detect vertices that are close to the edge of the image.** These vertices are excluded from the rest of the sequence and merely assigned as Al.
3. **Vertex mapping.** Assign all out-neighbourhood sets according to equation 3.10, and then all the other sets in the maps of the vertices, according to table 3.2.
4. **Zeta analysis.** Perform zeta analysis.
5. **Default model alpha prediction.** Using only the α_{\min} and α_{\max} attributes, query the active model for advanced species classification.
6. **Vertex mapping.** Update the vertex maps, identical to step 3.
7. **Zeta analysis.** Perform zeta analysis, identical to step 4.
8. **Precipitate detection.** Perform particle detection.
9. **Calculate normalized gamma.** Calculate the normalized peak and average gamma attributes.
10. **Weak untangling.** Perform weak atomic graph untangling according to the map in figure 3.9.
11. **Strong untangling.** Perform strong atomic graph untangling according to the map in figure 3.9.
12. **Zeta analysis.** Perform zeta analysis, identical to step 4.
13. **Default model prediction.** Using all the attributes of table 3.4, query the active model for advanced species classification.
14. **Vertex mapping.** Update the vertex maps, identical to step 3.
15. **Repeat steps.** Repeat the steps 8, 9, 10, 11 and 12.

3.4.4 A detailed example

The precise sequence presented in the previous section, section ??, is best illustrated with an example. This section will review the column characterization algorithm on a specific HAADF-STEM image, and present the state of the atomic overlay and atomic graph at each key step. Note that these atomic overlays will only show the atomic species, and not the advanced species of each column, and all though AutomAl 6000 can be configured to display advanced species overlays, the advanced species is mostly a service to the underlying statistics. The probability of a column belonging to a certain atomic species, is the sum of all the probabilities of each advanced species which maps to that particular atomic species.

The subject image of the example is presented in figure 3.10. Figures 3.11-3.21, tells a visual story about the interplay between the different constituent methods of the column characterization algorithm. Figure 3.22 shows the final result of the column characterization, while figure 3.23 shows the manually corrected state.

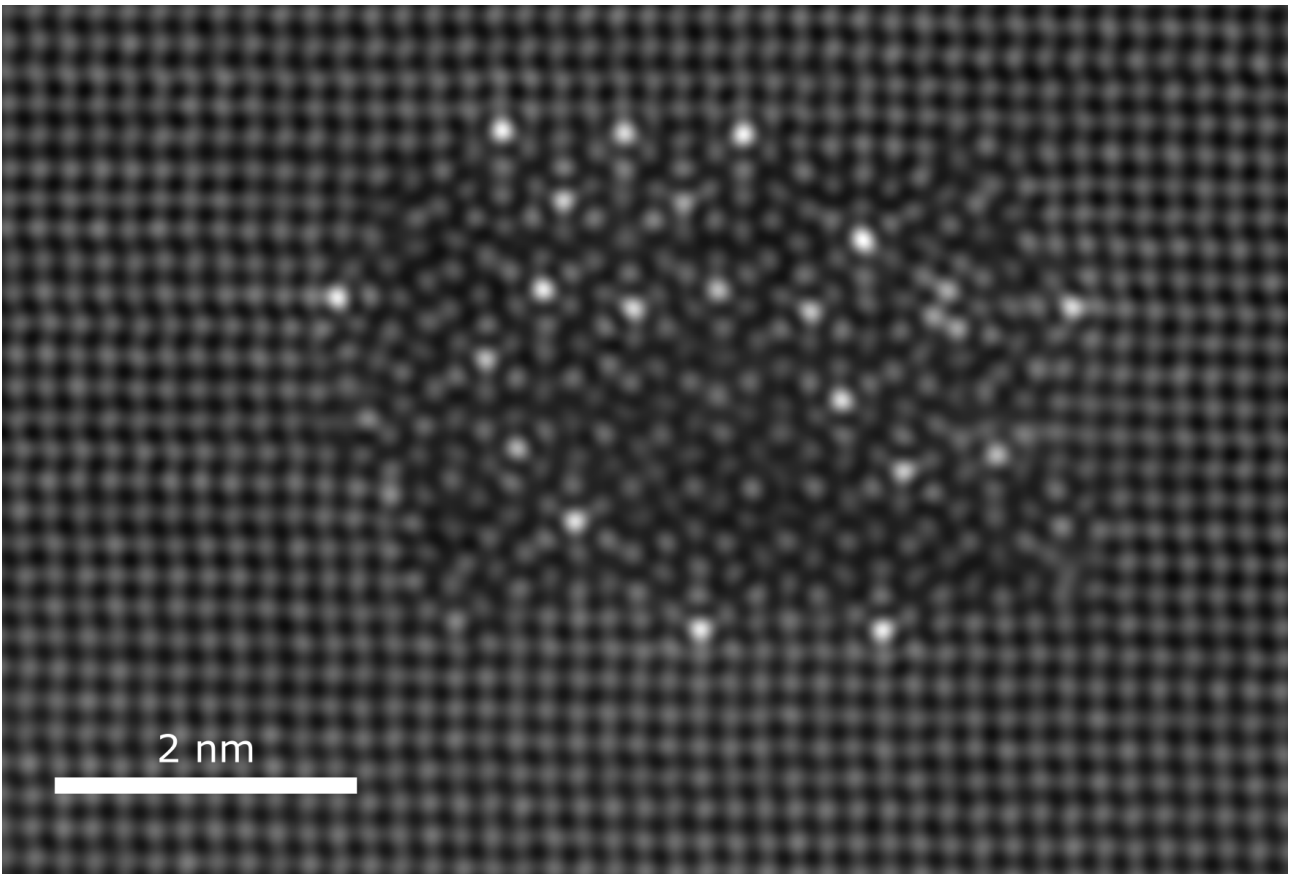
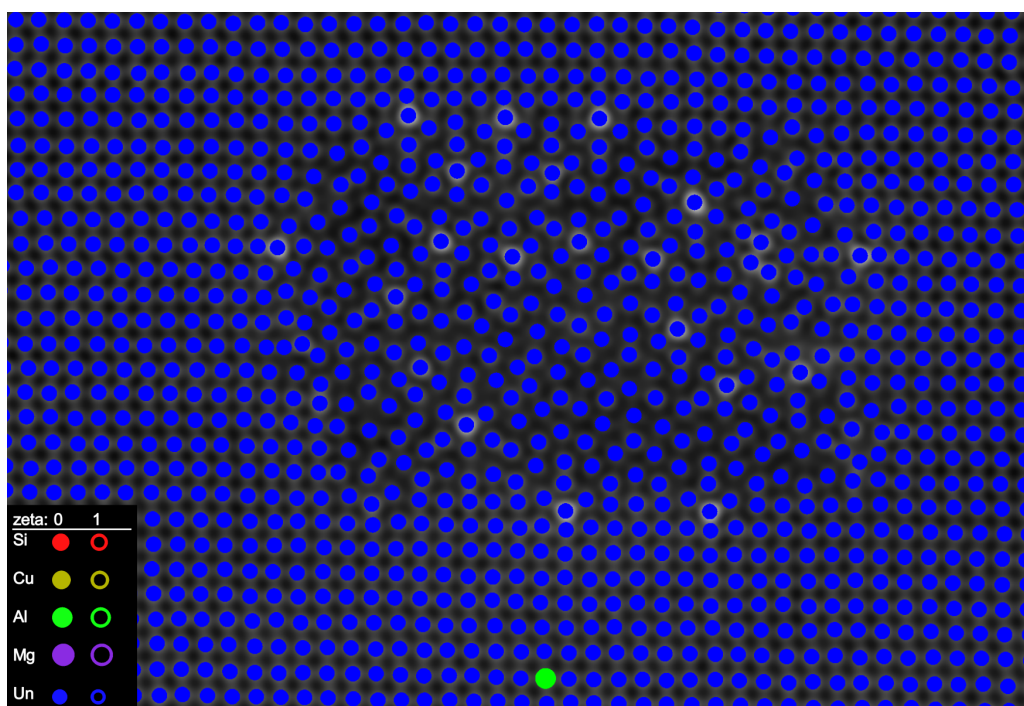
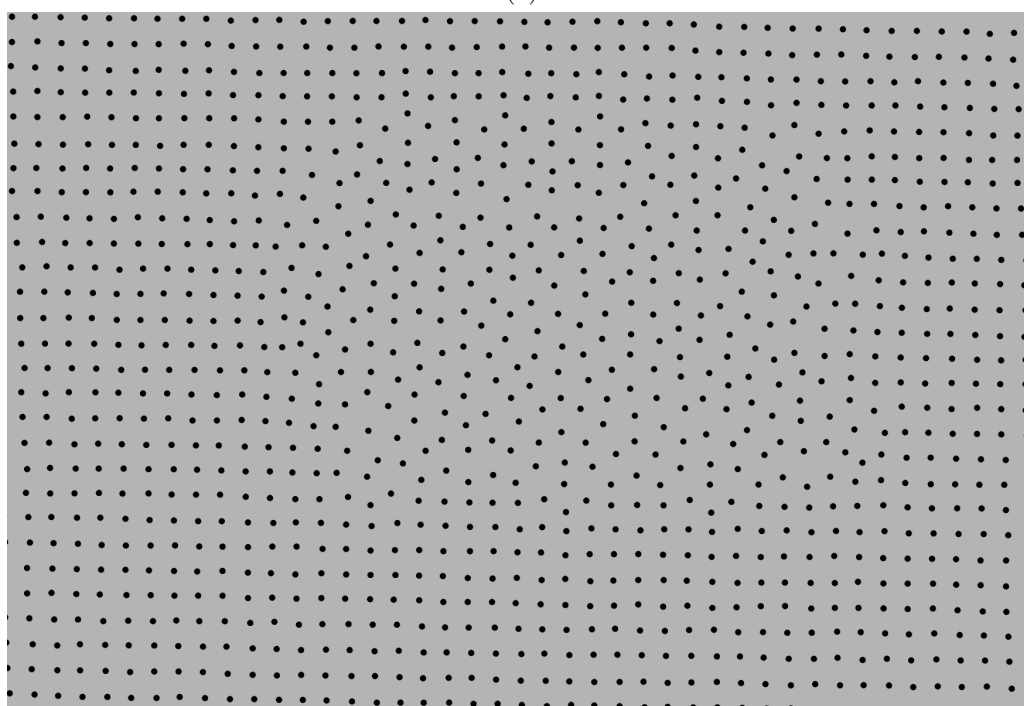


Fig. 3.10: A hybrid L and Q' precipitate.

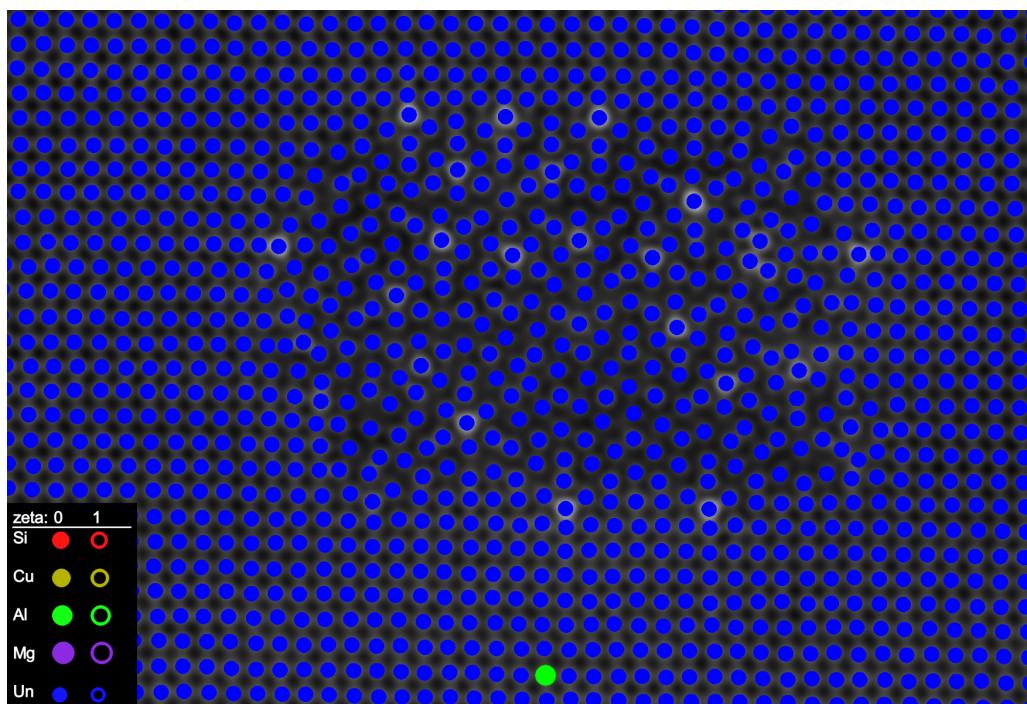


(a)

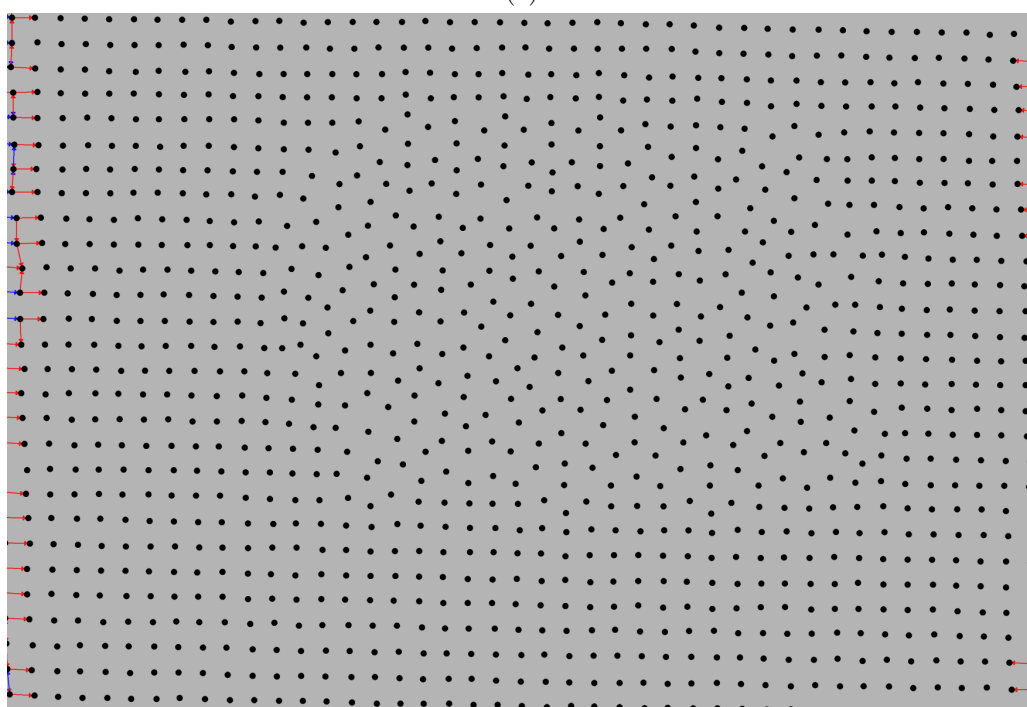


(b)

Fig. 3.11: The atomic overlay and atomic graph in its "initial" state, before column characterization is applied. (a) Atomic overlay. All columns are labelled as Un. Note the single matrix Al column, which must be set by the user and be the selected column when column characterization is initiated. (b) Atomic graph. No arcs are defined yet.

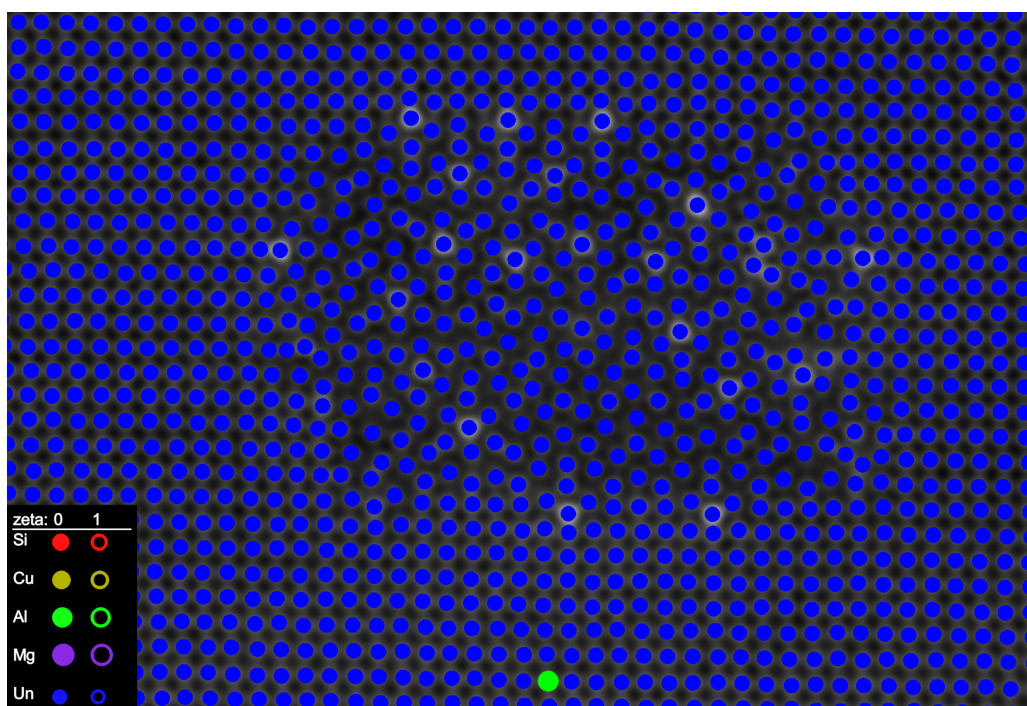


(a)

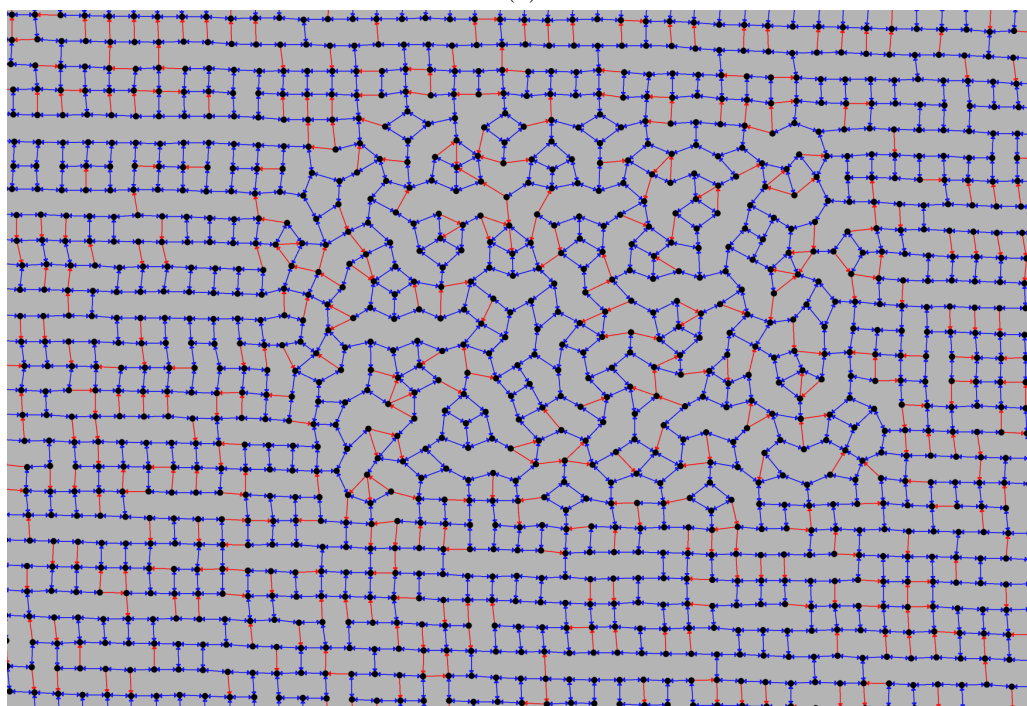


(b)

Fig. 3.12: The atomic overlay and atomic graph after step 1 and 2 of the column characterization algorithm. (a) Atomic overlay. This example image is cropped so that details will be visible, but this means that the edges of the overlay is not visible, where the columns are now set to Al, and will be disregarded for the rest of the algorithm. (b) Atomic graph. Even though districts are now assigned (step 1), the map of table 3.2 has not yet been applied.

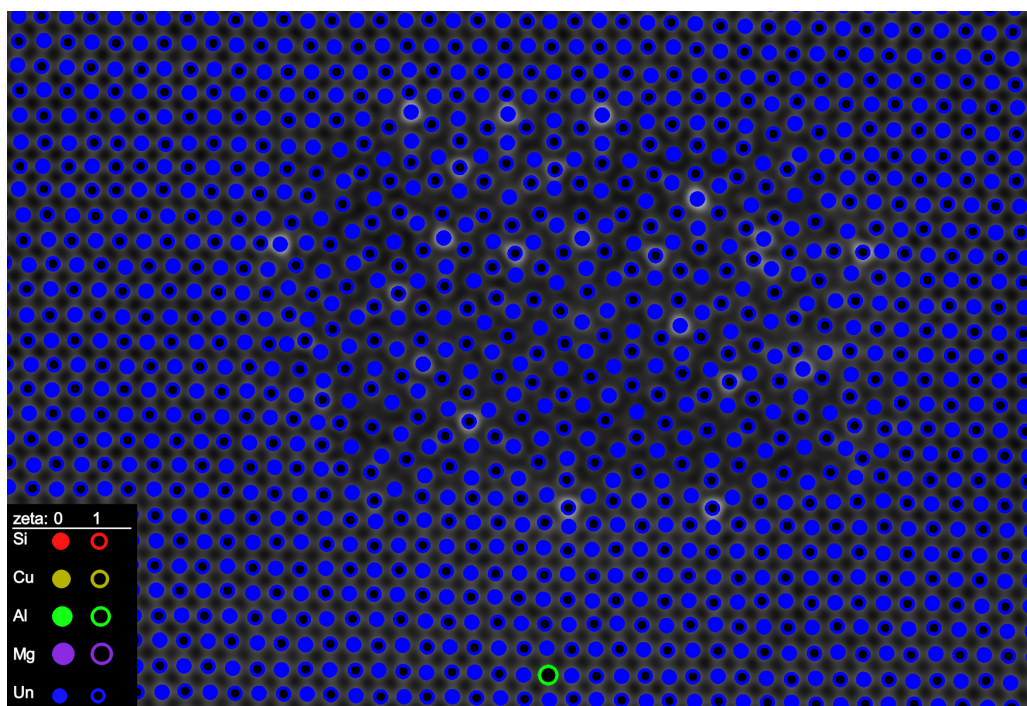


(a)

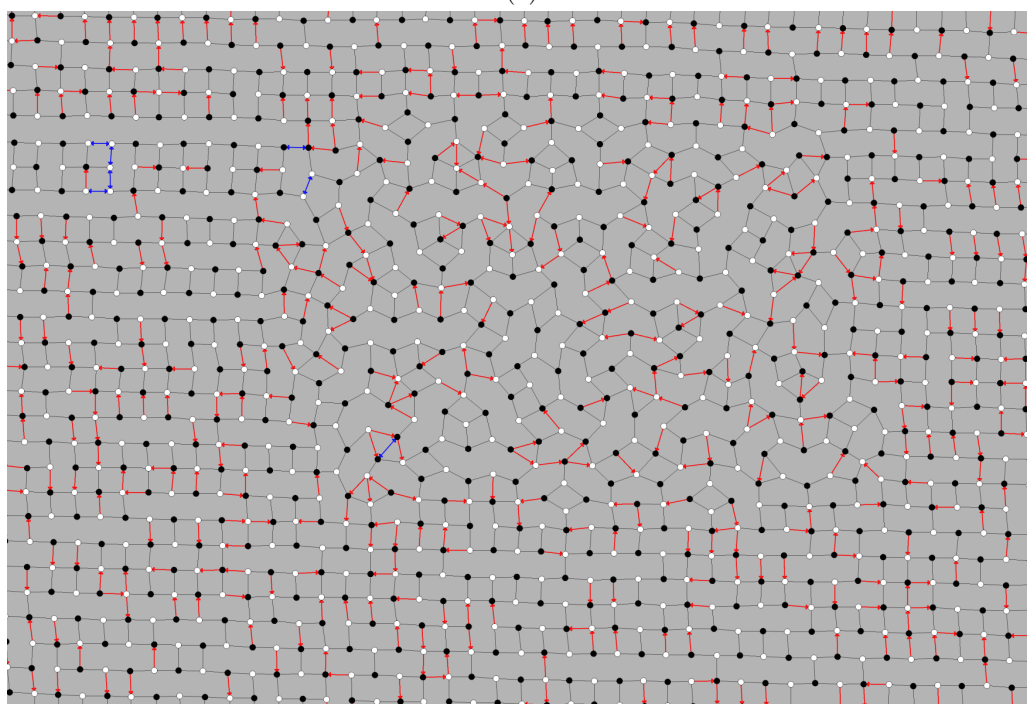


(b)

Fig. 3.13: In step 3, arcs are now assigned according to table 3.2. (a) Atomic overlay. No visible changes in the overlay. (b) Atomic graph. Since all vertices are still labelled as Un, they all have symmetry numbers equal to 3, which provides this peculiar patchwork pattern. Also, all vertices have zeta values equal to 0, so all symmetric arcs are blue.

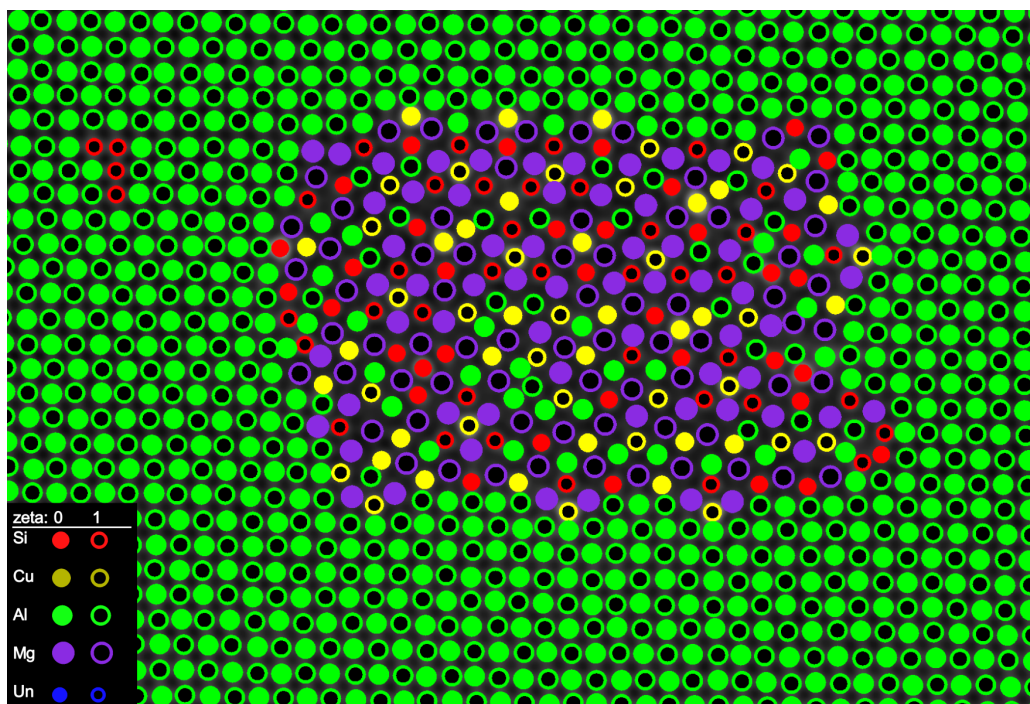


(a)

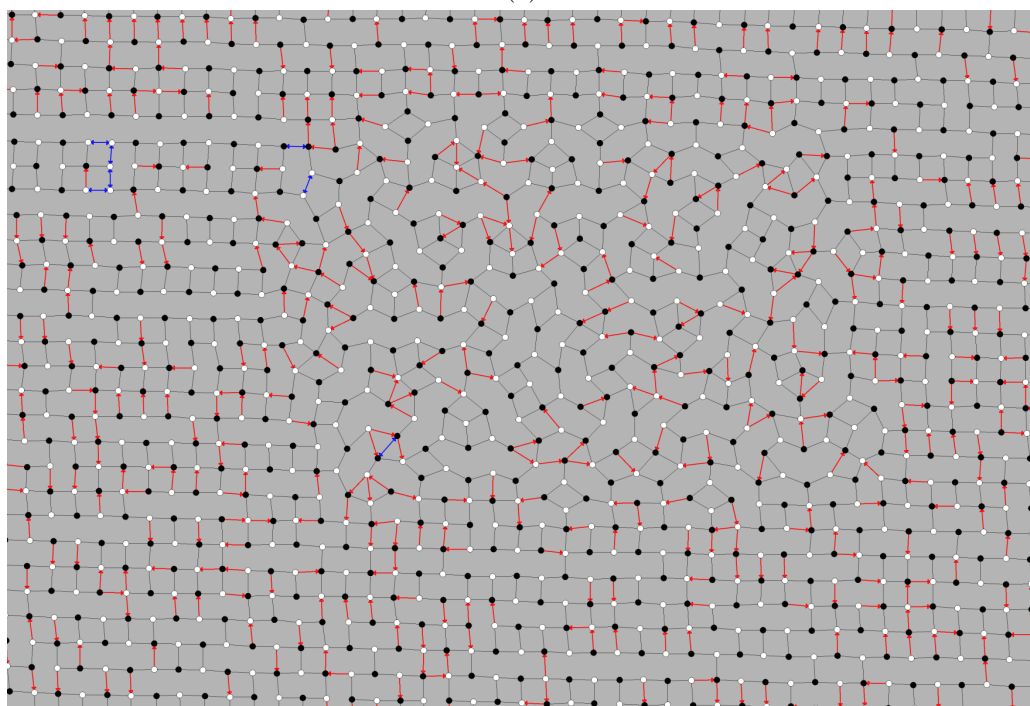


(b)

Fig. 3.14: In step 4, the first round of zeta analysis is performed. (a) Atomic overlay. The columns now have alternating zeta values. On this particular image, the zeta analysis seems to have performed well, as there are no visible "polarization" inversions across the overlay. (b) The atomic graph now comes alive. The thin black lines are symmetric arcs, which makes the un-symmetric and in-plane arcs stand out. There is very little blue in this graph, which is another indication that the first zeta analysis performed well.

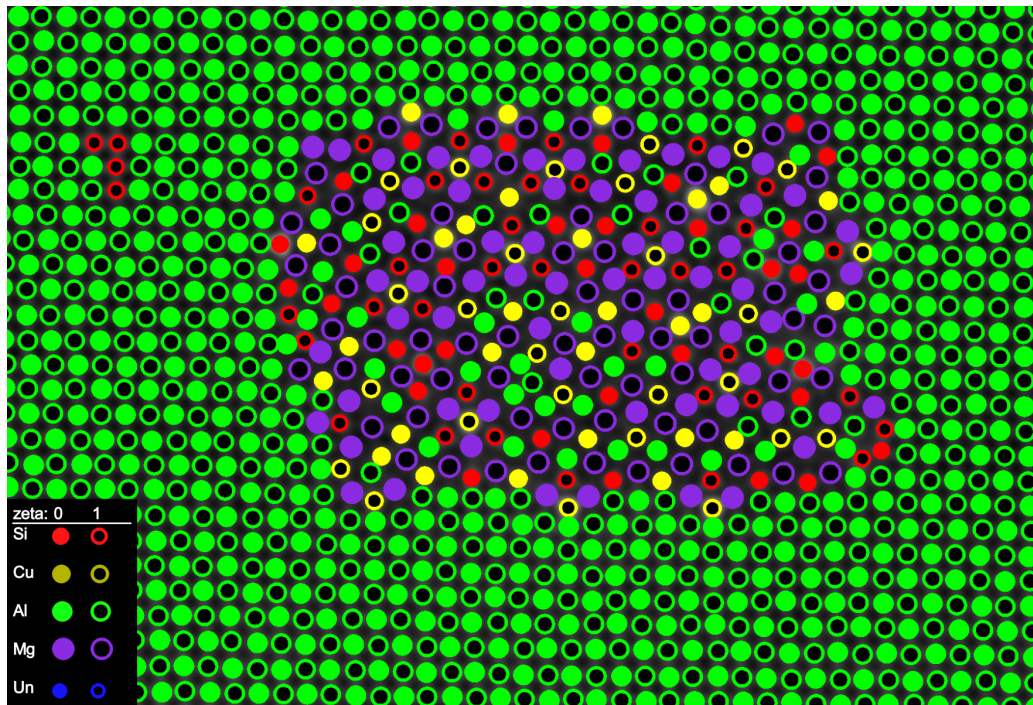


(a)

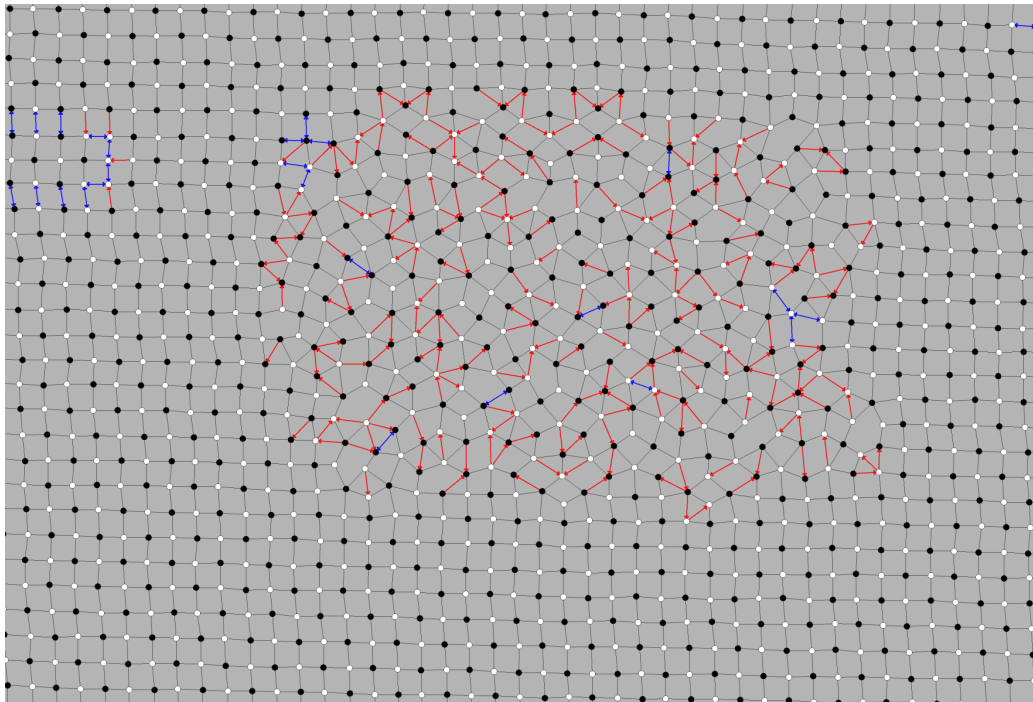


(b)

Fig. 3.15: In step 5, predictions based on the min and max alpha angles are made by the active model. (a) Atomic overlay. There are still many wrong predictions, but as can be seen, the alpha predictions are good enough to more or less correctly label the Al matrix. (b) The atomic graph will be identical to the previous step, because even though the symmetry numbers across the graph has now changed, the map of table 3.2 have not yet been updated.

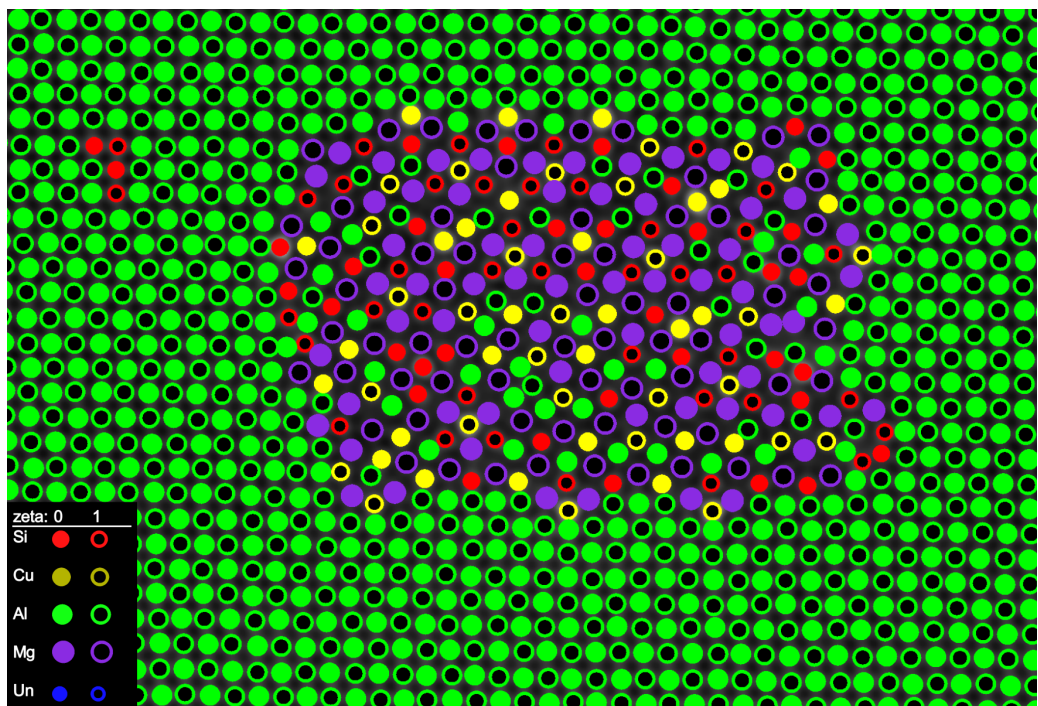


(a)

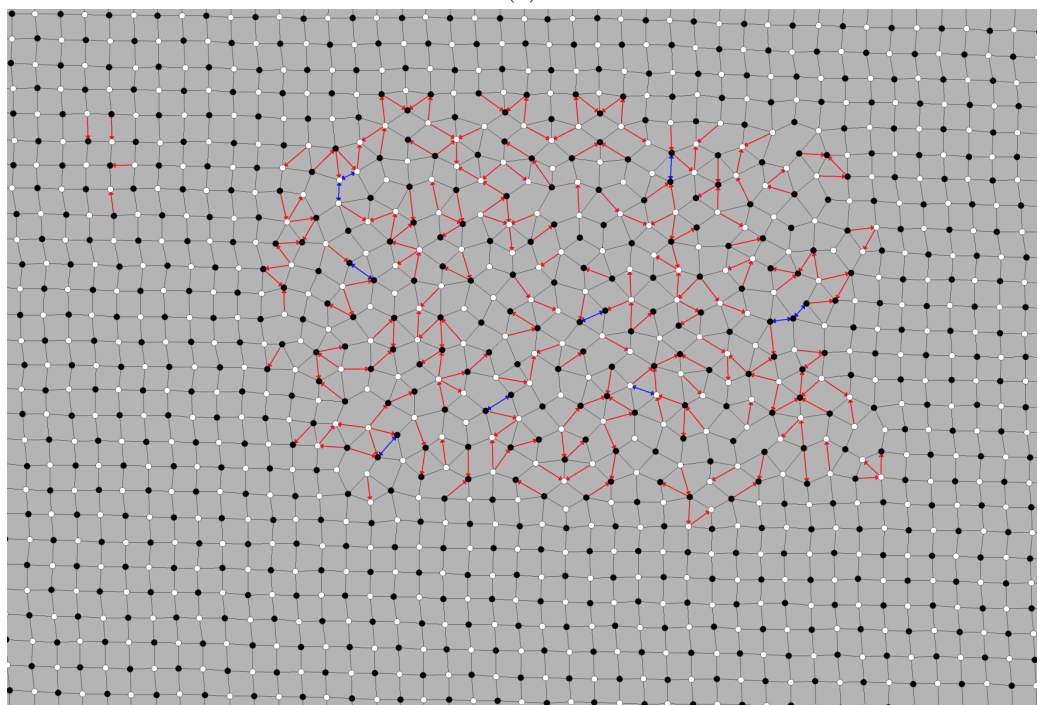


(b)

Fig. 3.16: Step 6 will reassign the arcs of the atomic graph according to the new symmetry numbers. (a) Atomic overlay, no change. (b) The atomic graph is now becoming more sensible. There are still many red un-symmetric arcs in the graph, especially in the precipitate, but due to the particular sequence, these are often predictable and will thus hopefully be effectively captured by the untangling map at a later stage.

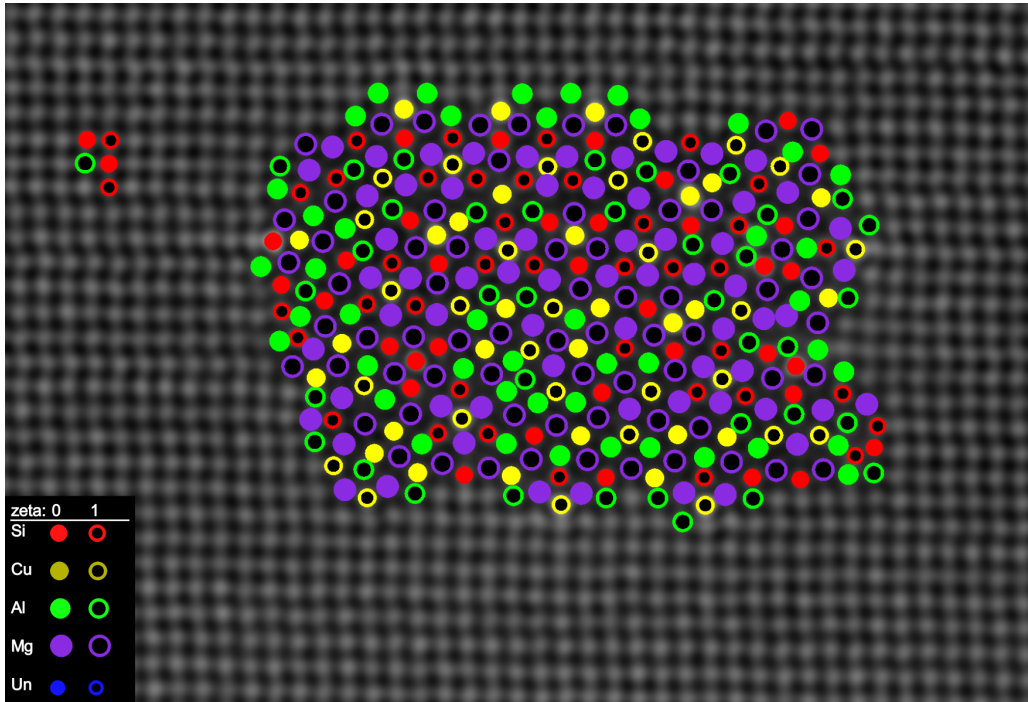


(a)

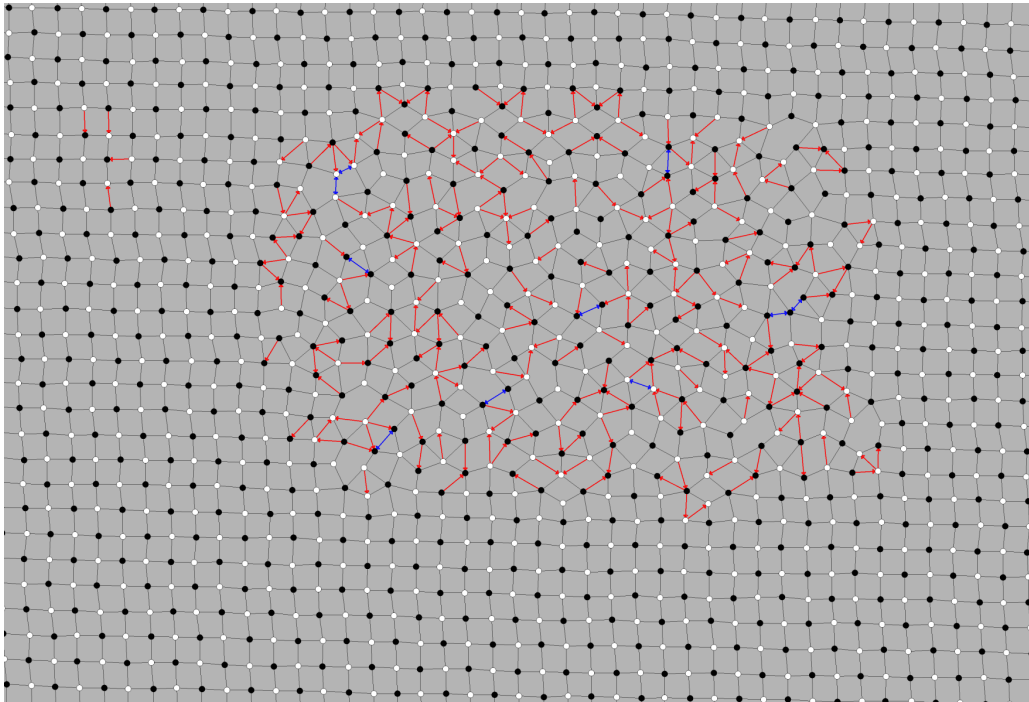


(b)

Fig. 3.17: Another round of zeta analysis takes place in step 7, since the graph information is now better, the zeta analysis will now perform slightly better, correcting some of the erroneous zeta labels. (a) Atomic overlay. (b) Atomic graph.

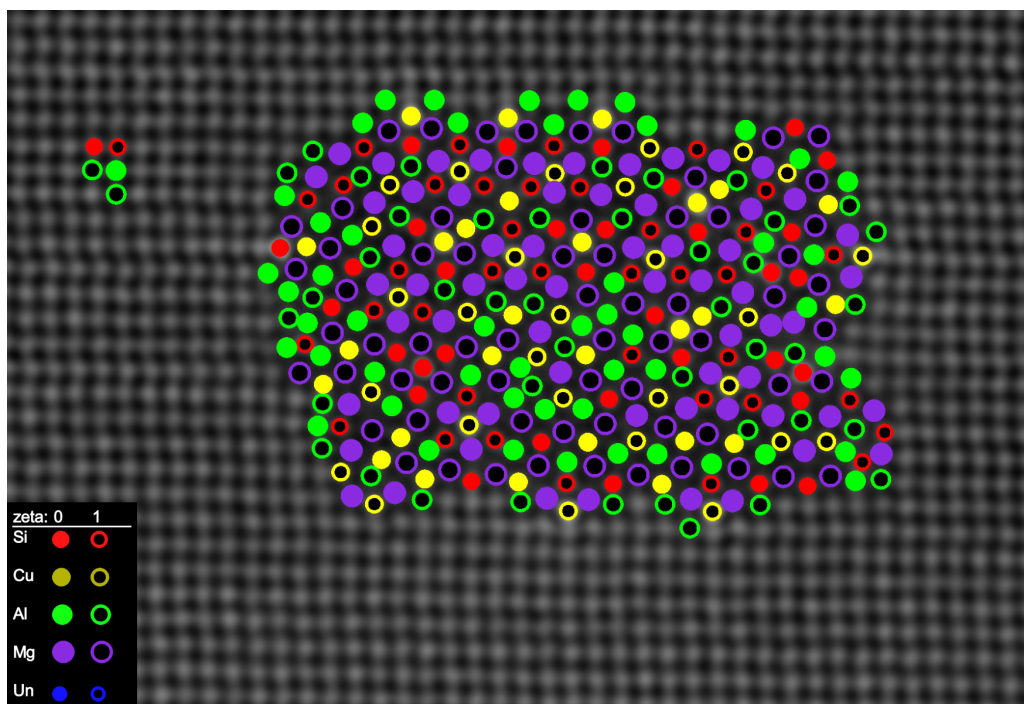


(a)

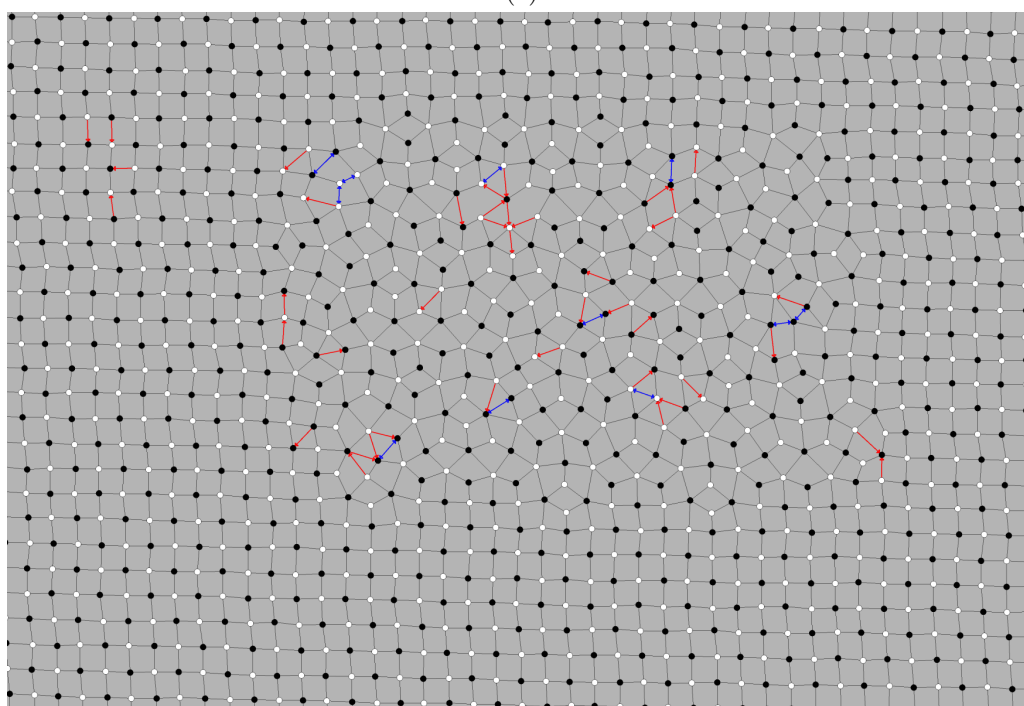


(b)

Fig. 3.18: In step 8, precipitate detection is performed. (a) In the AutomAl 6000 GUI, to show/hide matrix columns is one of many display options, which in this case is set to not display matrix overlay. At this point, the parts of the matrix that is correctly labelled, will not change during the rest of the algorithm, because all the information will only be supporting the current labelling, meaning that the atomic graph and overlay has already converged to its ideal prediction. Therefore the matrix is elected away from the overlay to declutter the overlay and bring forth the particle. (b) The atomic graph makes no distinction between particle and matrix vertices, so no changes at this stage.

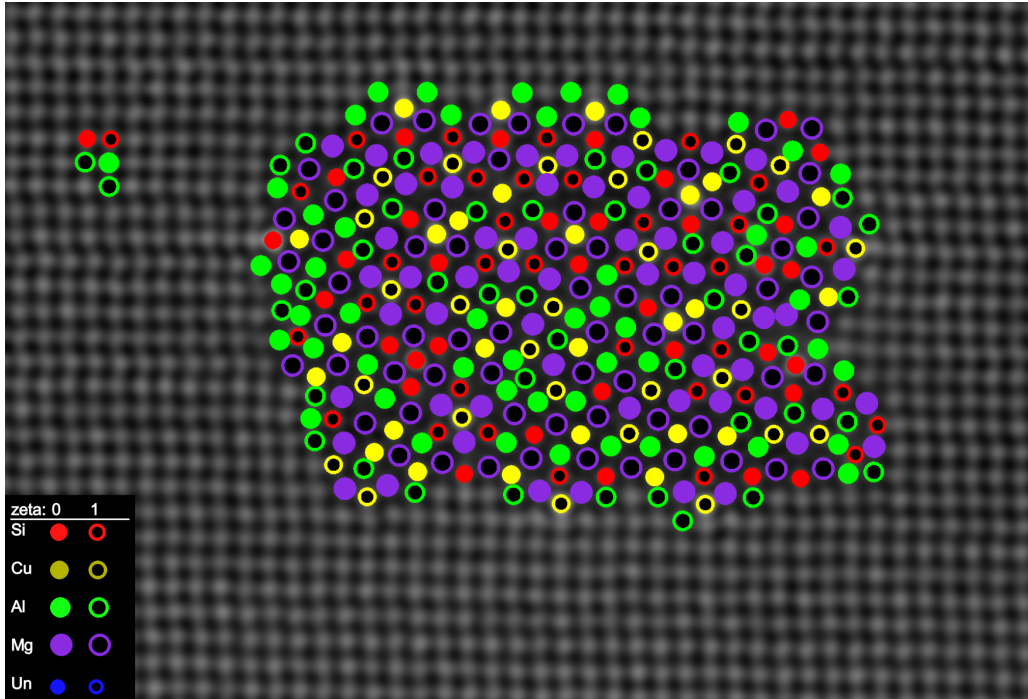


(a)

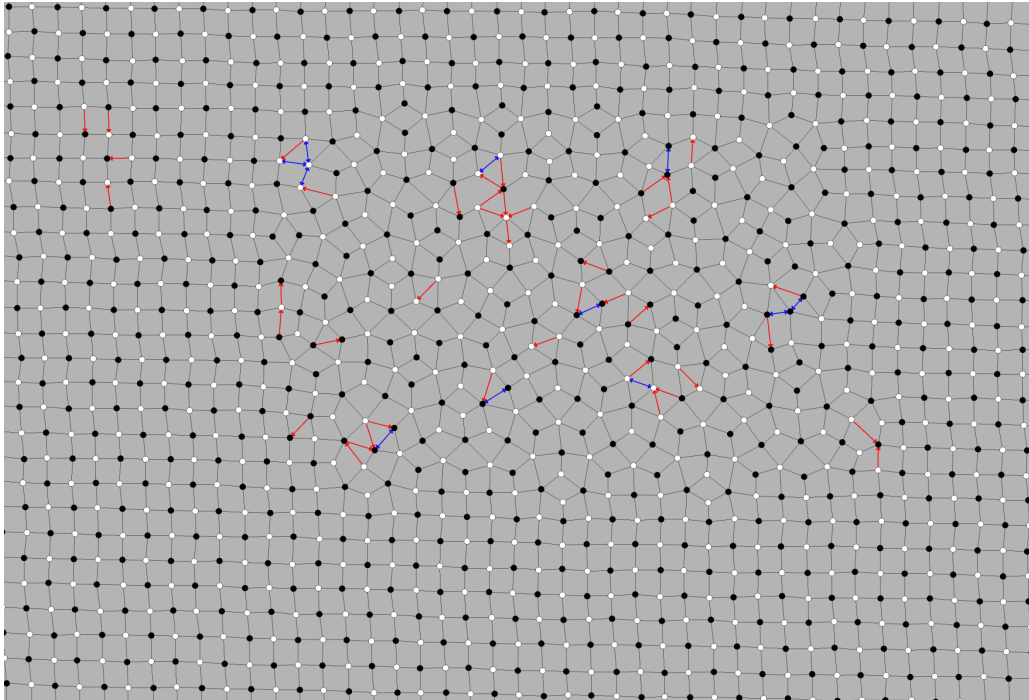


(b)

Fig. 3.19: Step 9 provides no visual indication in the overlay or graph, and this figure shows the overlay and graph after the sequence of steps 9, 10 and 11. That is, the transition from the previous figure (figure 3.18), to this figure, shows the effect of weak untangling, followed by strong untangling. (a) Atomic overlay. Untangling is designed to induce the minimal amount of changes to symmetries, and thus also to atomic species. Thus, the overlay is not expected to change very much, but we can for instance observe that strong untangling has correctly changed the atomic species of some of the matrix columns that was previously mislabeled. (b) The atomic graph changes dramatically at this stage, and now provides a much better informational representation of the actual atomic structure. The transition from the previous atomic graph, to this one, provides some merit to the use of the term "untangling" for this procedure.

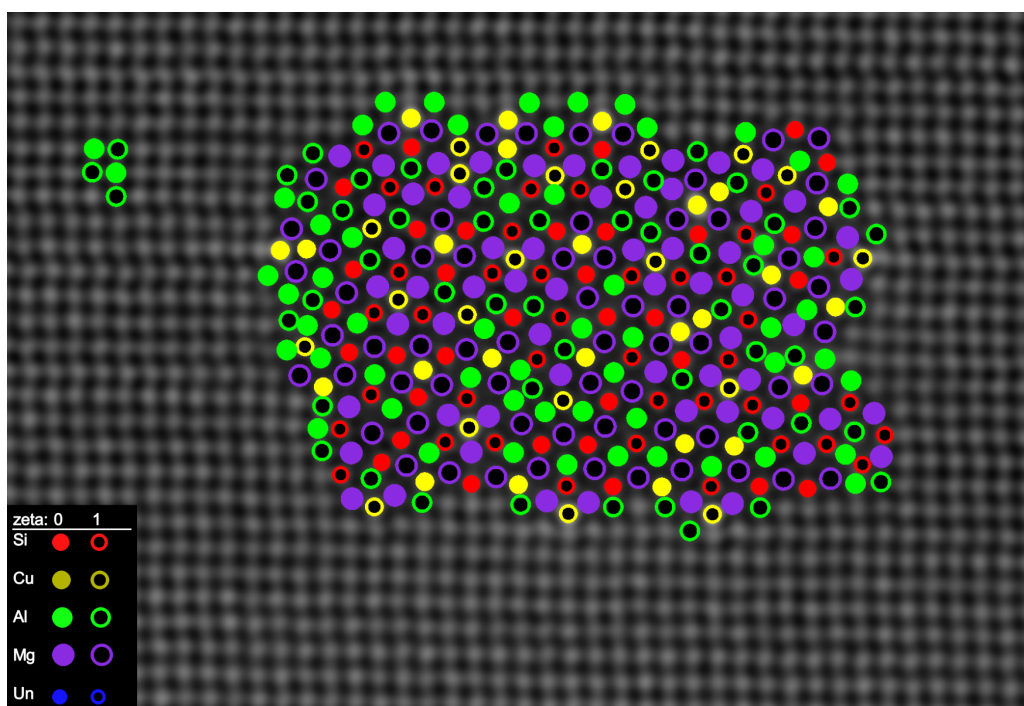


(a)

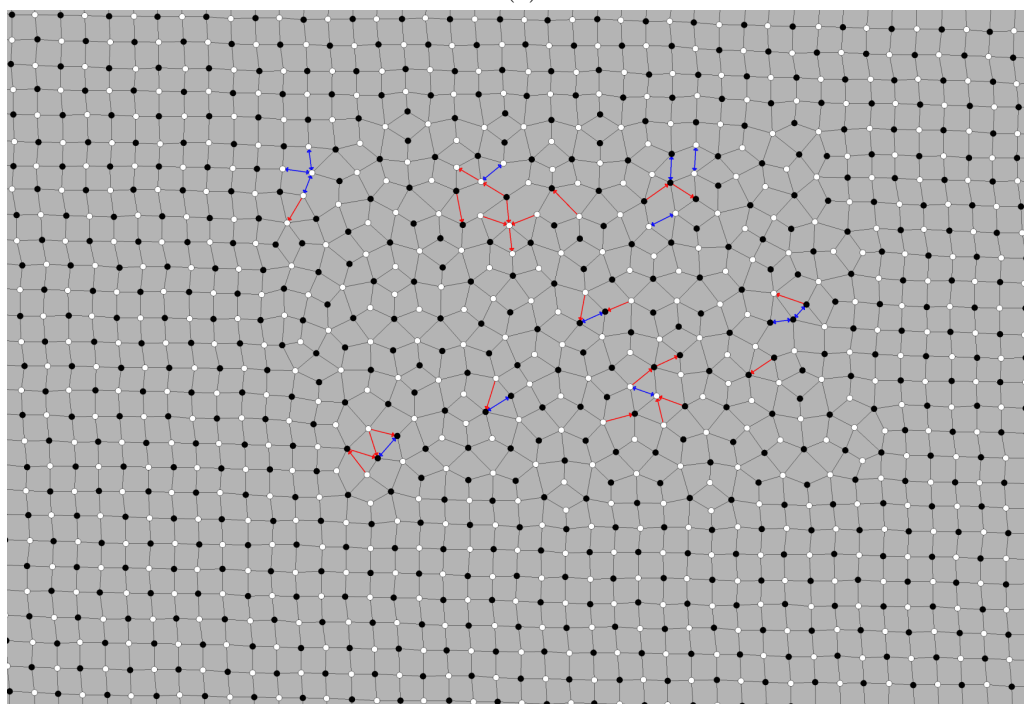


(b)

Fig. 3.20: Another round of zeta analysis is performed in step 12, using the now much more improved relational information of the untangled graph. (a) Atomic overlay. (b) Atomic graph. The voting system of the zeta analysis will only vote along symmetric arcs. The key for this to work, is that no vertex has more erroneous symmetrically adjacent vertices, than correct un-symmetrically adjacent vertices. The low density of un-symmetric arcs in this atomic graph, is thus promising for the effectiveness of the zeta-analysis.

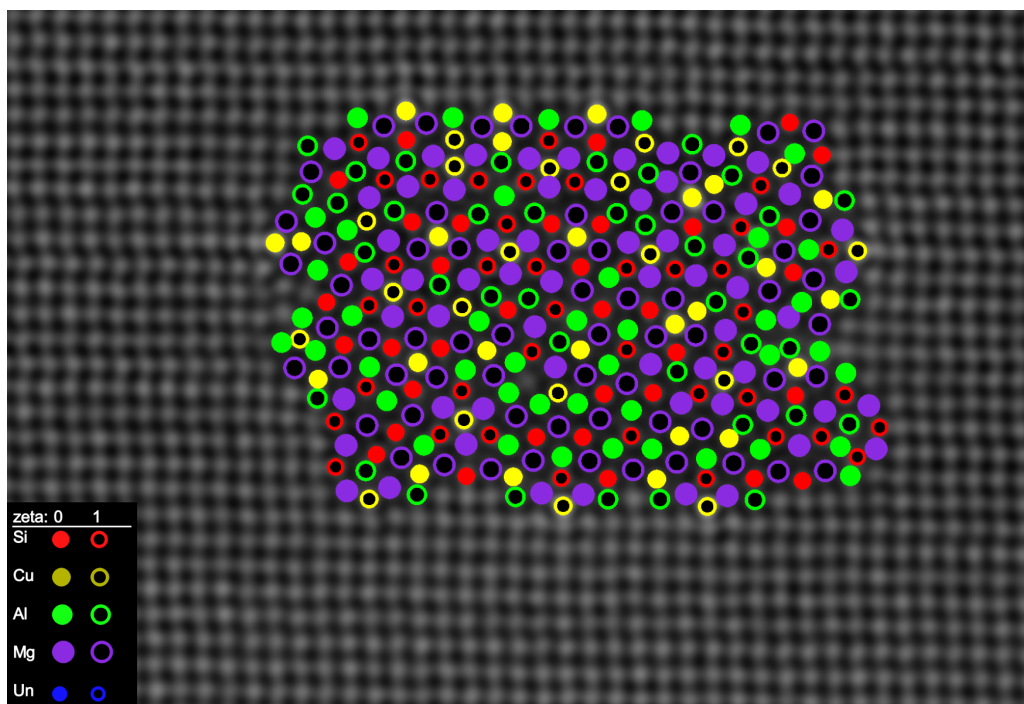


(a)

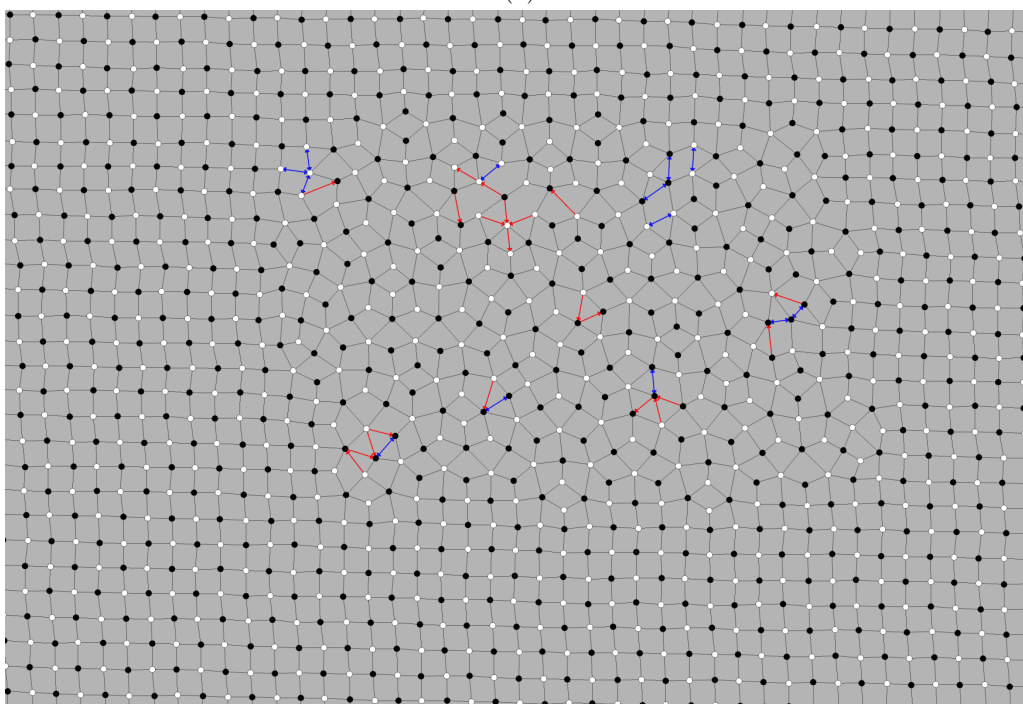


(b)

Fig. 3.21: Step 13 and 14. The untangled graph theta and alpha attributes, together with the normalized gamma attributes, are now fed to the active model for a prediction query that exploits the full strength of the model. The new symmetry numbers are then reflected in the graph by a new vertex mapping. (a) Atomic overlay. (b) Atomic graph.

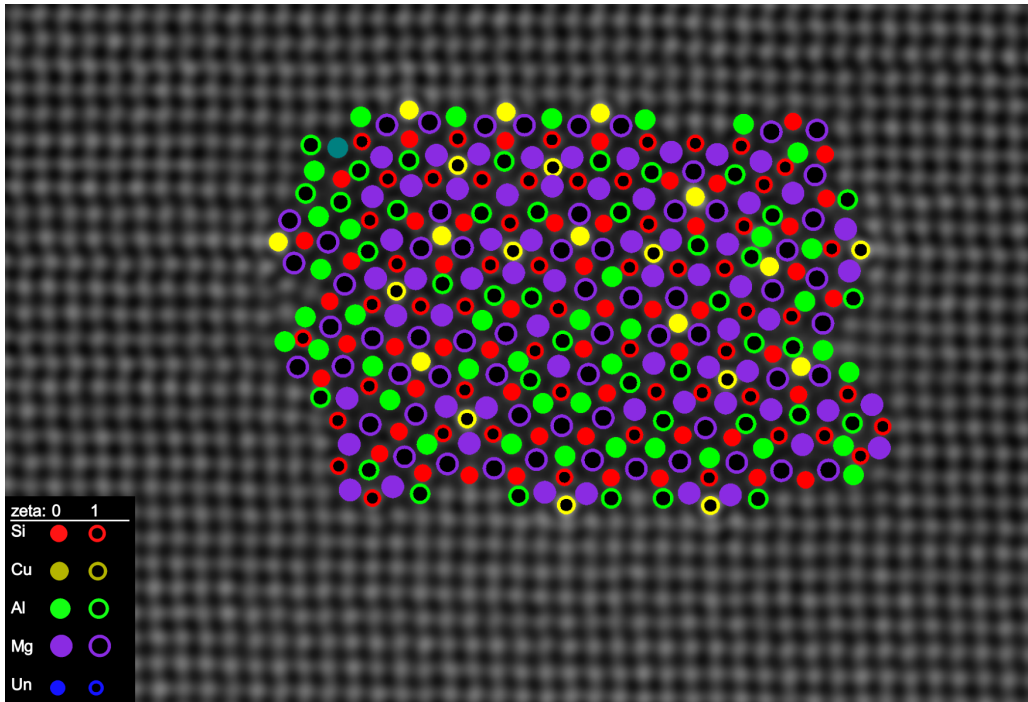


(a)

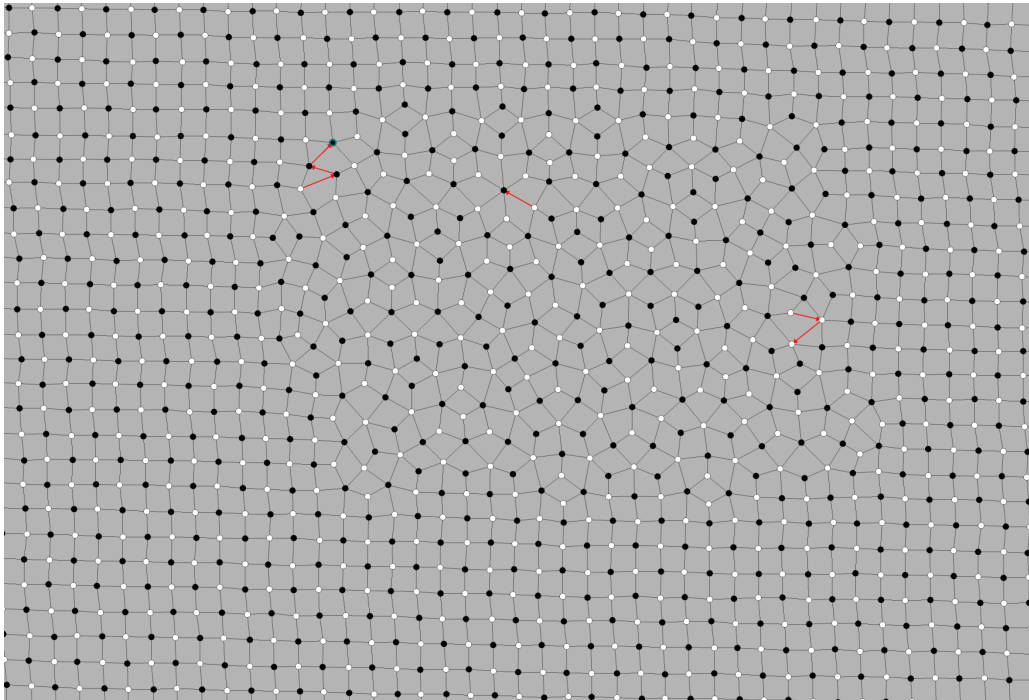


(b)

Fig. 3.22: Step 15, which repeats the steps 8, 9, 10, 11 and 12, will now produce the final result of the column characterization. (a) Atomic overlay. Some mislabeling of Si versus Cu is apparent. The matrix is correctly labelled, and thus not shown in the overlay. (b) The atomic graph still has some areas that stand out for the user to analyse. Much effort is put in to these methods such that the symmetric areas of the graph need not be closely inspected by the user, and can mostly be trusted as a correct analysis. The exception to this is the labelling of Cu versus Si. The atomic graph looks identical for these two species, because they have the same symmetry number. In this case, many of the Si columns are mislabelled as Cu, which is mostly on account of the default model.



(a)



(b)

Fig. 3.23: After the column characterization has concluded, manual inspection and correction of the result is performed for this figure. (a) Atomic overlay, with the now corrected Si/Cu columns. (b) Atomic graph. Despite now having been reviewed by an individual versed in atomic graphs and Al-Mg-Si-(Cu) precipitates, this is still not a perfectly symmetric graph. This is because, in some real-life scenarios, the structural principles of section 2.2.3, will not always hold, which is the case here. Note especially the un-symmetric arc in the top middle of the precipitate which indicates a vertex position with a symmetry number of 6. Since these are not included in the model, no vertex v_i can have $\deg^+(v_i) = 6$, but these 6-fold Mg positions do in fact seem to appear with some regularity in hybrid precipitates.

4 Discussion

This section provides some discussions on some of the most prominent topics that have come into light during development.

4.1 Column detection

Improving the column detection method of AutomAl 6000 could potentially improve several aspects of the software, like the accuracy of atomic positions, the effectiveness of the column characterization and/or the user experience by minimizing the amount of required manual intervention needed. Some specific suggestions on column detection improvements are discussed further in section 6.4.

4.2 Atomic graphs

Atomic graphs was developed as a necessity for the approach that was chosen for the methodology of this thesis. As such, the properties of these atomic graphs are strictly defined in service to the overarching goal of creating a successful algorithm for column characterization. However, using graphs in this way to describe atomic structure, also has some interesting merits. The first theorem of digraph theory, equation 2.7, is still valid on atomic graphs. Also, an atomic graph that represents a structure which complies with the principles of section 2.2.3, always has an average vertex degree of exactly 4, which can perhaps be linked to principle 5 in the atomic graph representation of the structural principles of section 3.2.4. Tying the graph current atomic graph descriptions back to actual physics, certainly seems possible. Precipitate number displacement (the number of columns in the precipitate versus the number of columns in the hypothetical FCC matrix which the precipitate displaces), packing fraction, polarization and possibly other properties, seems possible to link to the atomic graph properties. If the column characterization can be made more general then, AutomAl 6000 can possibly become a very advanced analytical tool, that can tell a lot about the image in a very short time.

There have been some other groups which seems to have deployed advanced graph theory to model atomic structures, of which notable mentions are [15, 16]. These papers have used undirected graphs to handle structure data, but the novel idea by the work presented herein, is to use digraphs to allow for a more flexible an algorithmic approach to discover the atomic structure. So atomic graphs might be a very purpose driven construct that has little use except the actual inner workings of the column characterization of section 3.4, but they might also have some more general usefulness, which is left as an open question here.

4.3 Column characterization

In general, the column characterization is not yet performing as well as it could. While the results are decent for many images, it leaves a lot to be desired for other images. An effort to make the method more general, robust and well understood, is an ongoing effort. The main conceived venues

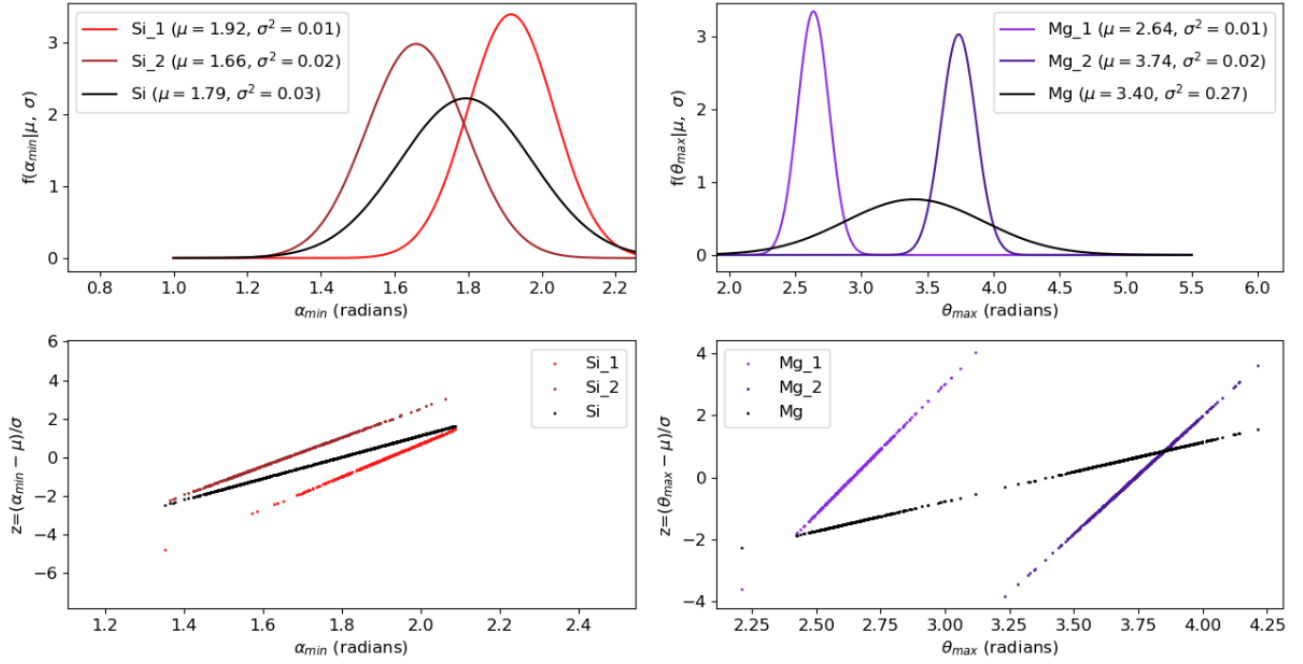


Fig. 4.1: Illustration of the complexion captured by the advanced species nominal attribute.

for improvements, lies either with the statistical model, the untangling or other as of yet unexplored methods. This section attempts to enlight some of these subjects further.

4.3.1 Statistical models

In section 3.3, the distributions of the default model was presented without much defense of its details. This section will examine the sample data a bit more closely. One of the complexions of the default model, is of course the advanced species nominal attribute. To illustrate a reasoning for this approach, examine the normal distributions of one single numerical attribute, the α_{\min} attribute. In figure 4.1a, the normal distribution of the Si atomic species nominal attribute, is compared against the normal distributions of the advanced species nominal attributes, Si₁ and Si₂. This "stretching" in the symmetry of the physically different structural functions, are captured by the separation of the means between the advanced species attributes. In figure 4.1b, the other example which was mentioned in section 3.3, is illustrated. Here, the separation of the means in the advanced species nominal attribute, is not a physical one, but a consequence of the definition of alpha angles, which can be predicted by examining figure 3.5. Figure 4.1c and 4.1d shows the corresponding normal probability plots, which is relevant to the following paragraph.

Once appropriate nominal classifications have been determined, another assumption of the default model, is that each numerical attribute is normally distributed within each nominal category. A check for normality is the normal probability plot, where each data point in the data is plotted against its z-score. Figure 4.2 shows the normal probability plots of each individual nominal and numerical attribute, and reflects the data that is used to calculate the distributions in figure 3.8. All though the attribute plots mostly fall on straight lines, which is a positive test for normality, the densities along each lines, seems to be slightly uneven. It is left as an open question here, whether there are

other distributions that would better model the numerical vertex attributes of the atomic graphs. In particular distributions that are not symmetric about the mean.

Some of the numerical attributes can be considered in pairs in scatter plots, which sometimes illuminate effects that disappear when data is "black boxed" as just "data". Figure 4.3a shows a scatter plot of the α_{\min} and α_{\max} attributes, with the normal distributions calculated from the projection of the data on each axis. Individually, these normal distributions does not reveal the peculiar shape of the data, which is a result of the definition of the alpha angles. There are two sharp boundaries through this 2D data, which can best be explained by considering what a data-point outside these boundaries would entail. Take for instance a hypothetical vertex which had $(\alpha_{\max}, \alpha_{\min}) = (3, 5, 1, 5)$. Both of these values are reasonably within the bell curve, so that the vertex could be considered an Al column for instance. But with these values, the "middle" of the alpha angles would be $2\pi - 3, 5 - 1, 5 = 1, 28$, but this "middle" angle is now smaller than the assumed minimum angle, which means that such a configuration would be mapped into the data-point $(3, 5, 1, 28)$. The maximum and minimum theta angles can be plotted in a similar manner, which is shown in figure 4.3b. Figure 4.3c shows a scatter plot of the normalized peak gamma versus the normalized average gamma. This plot illustrates the effect of images that are "overexposed" in the data.

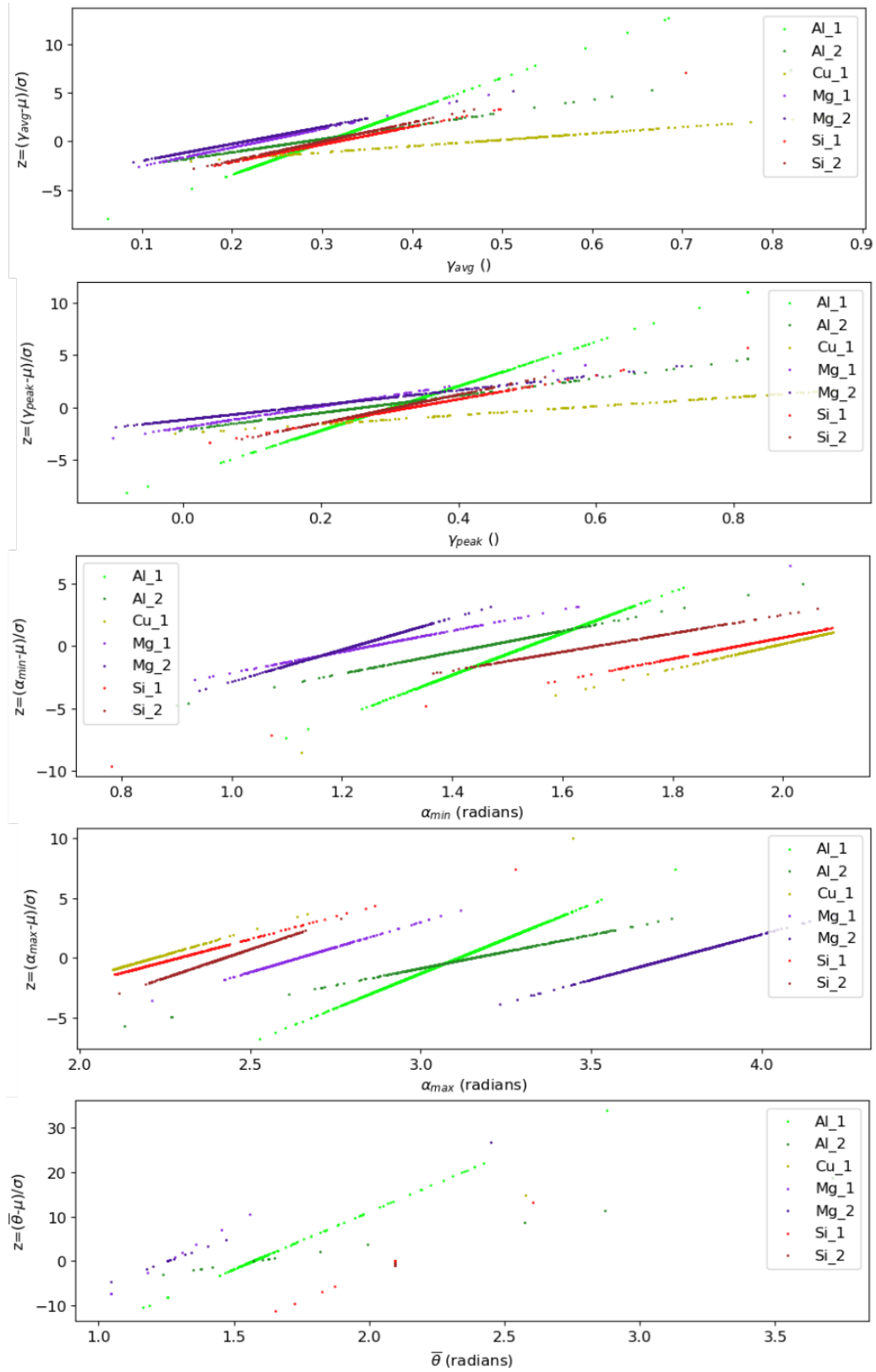


Fig. 4.2: The normal probability plots of the default model corresponding to the attributes of figure 3.8.

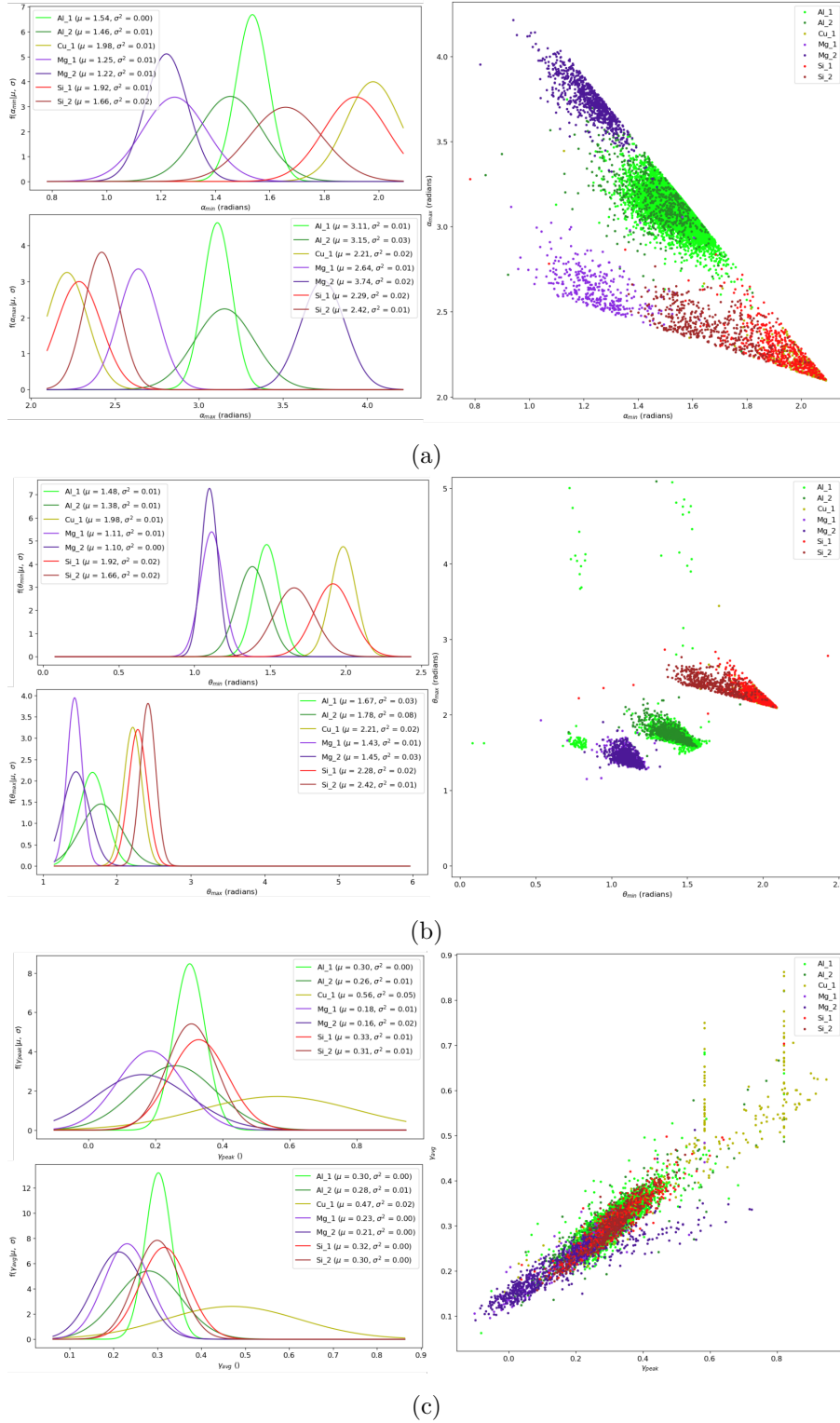


Fig. 4.3: Pairs of related numerical attributes of the default model. (a) Scatter plot of the α_{\min} versus the α_{\max} attribute. The plots on the left also show the single attribute normal distributions. (b) Theta angles. Note the separation between Si₁ and Si₂. (c) Scatter plot of γ_{peak} versus γ_{avg} . Notice how the overexposed images breaks the linear correlation between peak and average gamma. This also causes the variance of the Cu₁ curve to be a lot less sharp (low variance) than it could have been. A better data sample, would create a better model.

4.3.2 Zeta graphs

In section 3.2, it was hinted at more than one conceivable way to select the members of the out-neighbourhoods from the districts. One such way, given a vertex v_i , is to select the first n_i vertices v_j where $\zeta_i \neq \zeta_j$. In set-builder notation then, let

$$N^+(v_i) = \{v_j \in D(v_i) | \zeta_j \neq \zeta_i \wedge |N^+(v_i)| < n_i\}. \quad (4.1)$$

If there are not enough vertices in $D(v_i)$ to fill $N^+(v_i)$, then repeat the process by adding the first vertices v_j that are not already in $N^+(v_i)$. When the out-neighbourhoods are defined in this way, with all the other sets of table 3.2 defined in the same way as before, assuming the out-neighbourhood of equation 4.1, then let the resulting graph be the *zeta graph*.

The strength which the zeta graph has over the regular atomic graphs, is that given correct symmetry numbers and zeta values for each vertex, the zeta graph will almost automatically be the correct representation of the atomic structure just from its definition, without any need for untangling or other advanced graph analysis. This is in stark contrast to the atomic graphs, which can appear densely un-symmetric and still have correct symmetry numbers and zeta values. This can be demonstrated with a reverse-engineering example. Take a precipitate and by some method assure that it is the correct overlay of the precipitate. Then reset the atomic graph by re-running the spatial mapping of section 3.4.1, which will reset the vertex districts. Then re-apply the definition of table 3.2 to make the atomic graph. In a different instance, apply the zeta graph definition to the same situation and compare the results. This demonstration has been performed to produce figure 4.4.

Now, the zeta graph requires correct symmetry numbers and zeta values, but as it turns out, if at minimum the zeta values are correct, the zeta graph will still be a more simple object to extend the algorithm on, than the atomic graphs. So, if the zeta analysis method of section 3.4.1 can reliably assign correct zeta values a-priori, then this would be a much better, faster, accurate and reliable approach, than the untangling method. However, if the zeta analysis does not perform well, and some inversion occurs across the image, then this approach fails. This potential path of improvement with zeta graphs, hinges on how sophisticated the zeta analysis can become, which is left as an open question here.

Even if a column characterization based on zeta graphs will not bear fruits, the zeta graphs are still a useful tool as an interface between AutomAl 6000 and its user. The review of an atomic graph, requires the user to perform graph permutations on the graph, so as to manually untangle whatever problematic areas that remains. If it is a "difficult" image, this review can sometimes be a bit hard and demand some time and effort. However, if working in the zeta graph, all permutations are automatic, which means that the user must simply set correct symmetry and/or zeta values, which can be done at the push of button, and is thus much faster. Unless the zeta graph is too far removed from the actual situation it is supposed to represent, and once one gets used to the behaviour of the zeta graph, it provides a much smoother interaction with the AutomAl 6000 data-structure.

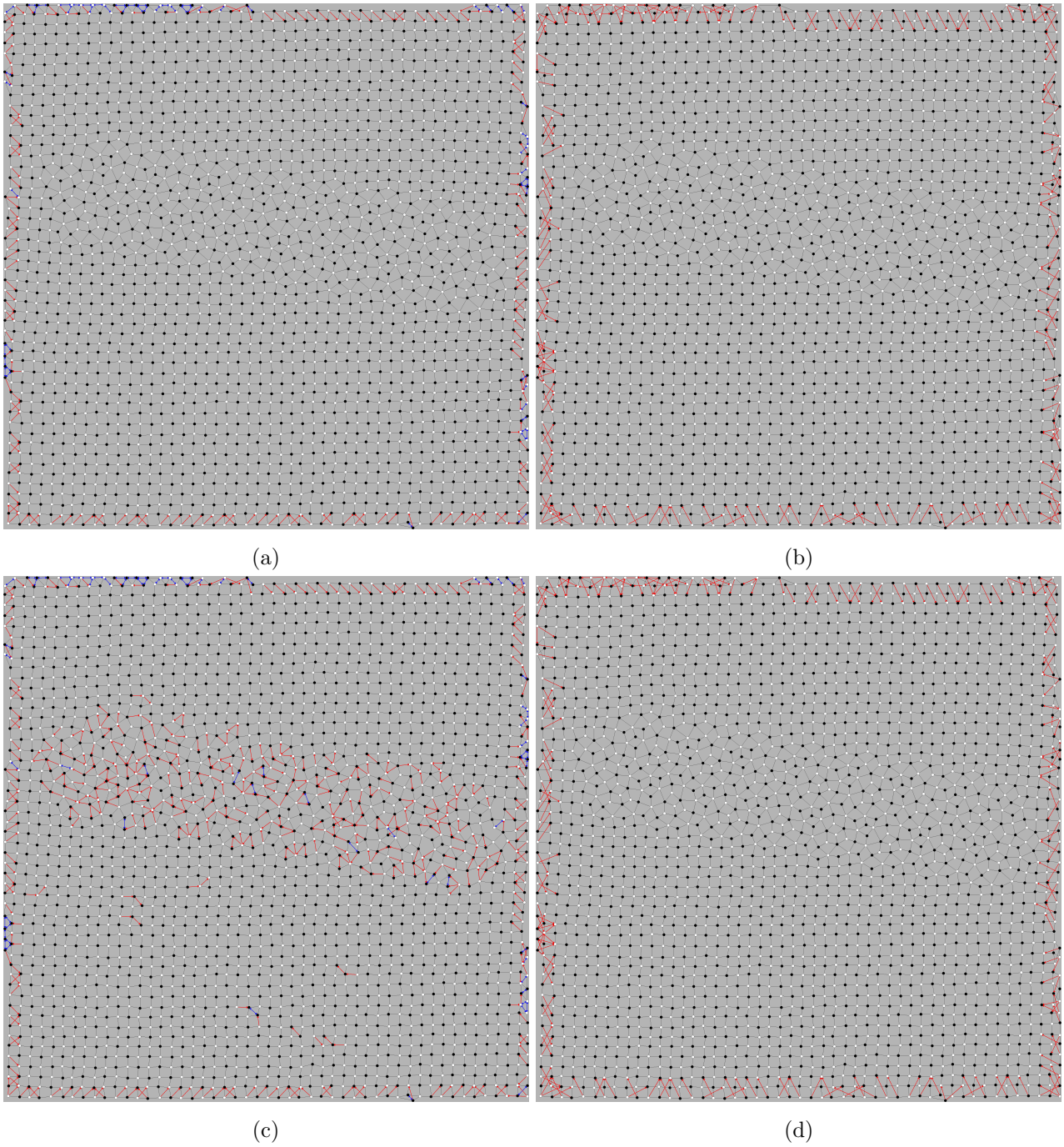


Fig. 4.4: Comparison of the atomic graph and the zeta graph after a district reset with the spatial mapping method. (a) The correct atomic graph of a Q' structure. (b) The correct zeta graph of the structure. (c) The atomic graph after reassigned districts according to the spatial mapping method, and the definitions of table 3.2. Note that although every vertex symmetry number and zeta value are correctly labelled, the atomic graph has a very low symmetry density in the precipitate. (d) The zeta graph after reassigned districts according to the the spatial mapping method, and the definition of the zeta graph. Note that the zeta graph is unchanged by the reordering of the districts, and remains the correct graph given correct symmetry numbers and zeta values.

4.3.3 Algorithms versus machine learning

Machine learning is a procedure that is swiftly becoming a common tool when approaching difficult problems. The method is concerned with constructing a *neural network*, much like a weighted graph, which will produce some output for a given data input, see for instance [17]. The range of problems that machine learning is applicable to, is vast. One of the requirements for machine learning, is that it requires a lot of data, appropriately configured, labelled and formatted for the training, which is most commonly done by gradient decent on the entire space of all the parameters in the graph. And even with a large set of training data, neural networks will typically suffer from diminishing returns after a certain point. This entails that by doubling the available training data, one does not necessarily double the accuracy which the neural network is able to achieve, thus producing a theoretical roof on how well the neural network can perform. Some approaches to atomic resolution image analysis using neural networks has already been done, notably [18, 19]. One advantage with an algorithmic approach though, is that developing an algorithm will offer much more insight into what is going on. All though neural networks can be analysed in many different ways, how it actually works can often be totally or partially encrypted in the sometimes vary large neural networks (millions of parameters in some cases). In the future, databases of overlays, conceivably created with AutomAl 6000, could potentially become large enough to enable machine learning as an alternative or symbiotic approach.

5 Conclusion

When a researcher is manually creating an atomic overlay of an Al-Mg-Si-(Cu) precipitate by using the structural principles, he or she will in general not consider this a difficult task, albeit a tedious one. Excluding difficult and/or unusual areas, and given reasonable familiarity with the 6000-system, the overlaying process is fairly straight forward, cognitively speaking. To translate this menial task into an algorithm, has turned out to be not so simple though. There are sometimes many advanced concepts and hidden layers in the cognitive process when a human applies its "intuition", "understanding" and/or "experience". The translation of the overlaying process into an algorithm has required the application of statistics, graph theory, specialized methods and a delicate interplay between the three. It requires significant effort from AutomAl 6000's algorithms to become confident of the symmetry of a column, that is, whether it is 3, 4 or 5. In most simple cases, a human can recognize this instantly, seemingly without much effort.

The resultant algorithm presented in this thesis is probably not the ultimate approach to this problem, but even if it can not outperform a human in accuracy, it can of course outperform a human in speed. The required manual review of the algorithm outputs, will typically require a lot less of a researcher's time, than manual overlaying, and this manual revision is amply aided by the visual simplicity of the atomic graphs. There is also an inherent strength in having the atomic structure ordered in a deliberate data structure, as it can be queried in almost any conceivable way. Manually counting atoms to determine the composition of a particle should no longer be necessary, and novel statistical insights seems to be within reach, perhaps through a systematic application of AutomAl 6000 at the hands of an adept researcher. Examples of studies that potentially could have benefited from AutomAl 6000, either for the purpose of extracting composition statistics from images or to create atomic overlays, are plentiful [20, 21, 22, 23, 24, 25].

6 Further work

Some work still remains, before AutomAl 6000 can be confidently applied as an exploratory research tool. Some of the most pressing issues that remain, as well as some of the most promising ideas, are presented in this section.

6.1 Technical docstrings

A docstring is a string of text that appears in the preamble of a method or class, that clarifies the arguments, exceptions, purpose and special considerations of the method or class. Proper docstring documentation is an important hallmark of quality code. At the time of writing, vast sections of AutomAl 6000's source code lacks docstrings. Other parts may have docstrings that are outdated or inaccurate. Time restraints has made it impossible to document the source code fully in time, but this takes highest priority going forward. Giving away AutomAl 6000 to the community as an open-source project, would be an empty gesture, if others where unable to contribute or develop AutomAl 6000 further due to a vertical learning curve into its current implementation. The current docstring format used in the source code, follows the Sphinx style guide [26]. If the docstrings are properly formatted in accordance with any compatible style guide, software like Sphinx can produce HyperText Markup Language (HTML) files which can be used to make online documentation, without having to actually write HTML. This has become the standard way to document python projects¹⁹.

6.2 Exception protection

The source code for AutomAl 6000 systematically lack any exception handling, even in critical areas. This could potentially be a source of frustration for users, due to unexplained and frequent crashes and bugs. If AutomAl 6000 is to be accepted as a useful tool by the community, making a stable version is of very high priority.

6.3 Additional export and import formats

At the time of writing, the data of AutomAl 6000 can be exported as a Comma Separated Values (CSV) file, which is a commonly used file format for parsing tabular data between different software. There are however some additional export formats that would be highly useful in this case. In particular, the Scalable Vector Graphics (SVG) file format, which would allow users to export the overlay data into a file format that could be edited with vector graphics software, such as Inkscape [27]. This would be highly beneficial, because overlay graphics could be exported with preserved layered data, which would enable researchers to generate publishing quality graphics from AutomAl 6000 overlays, which has rather limited graphical customization on its own. The SVG file format is an open standard with available documentation, and should therefore not be to hard to implement as an export format.

The current version of AutomAl 6000 can only import DM3 files, but there should of course also be some more options here.

¹⁹Citation needed

6.4 Improvements to column detection

Desired improvements to column detection includes a way to determine the ideal threshold value automatically at run-time, as well as a general improvement to its accuracy. Another option all together is to perhaps allow other open-source software that are specialized on column detection to take part in the AutomAl 6000 process at this stage, either by directly importing functionality from such modules into AutomAl 6000, or by crating a conversion tool that could convert file formats such that AutomAl 6000 would be able to read out the atomic positions. Atomap is a column detection software that is already widely in use by the community, open source and is rather sophisticated [12]. The lack of effort to develop advanced column detection for AutomAl 6000, is partly on the account that this problem is in some sense already solved. To conclude this short discussion, the column detection method as it is described in section 3.1.2, all though possibly adequate for a familiar user, is a dated minimal working naive conceptualization and should be considered for a partial or total re-work.

6.5 Improvements to column characterization

As was discussed in section 4.3.2, the zeta analysis together with zeta graphs, seems like an interesting venue of exploration for a more general column characterization method. Before this can be explored, a very robust version of zeta analysis must be arrived at. Another immediate improvements to the column characterization, would be to make more refined statistical models. In particular, the current version is not very good at distinguish between Cu and Si. A model build on a larger data set, where all overexposed images was removed, would decrease the variance of the Cu curves in the normalized peak gamma and normalized average gamma attributes, and thus improve the separation between Cu₁, Si₁ and Si₂. If the methods of the column characterization can be further generalized, it should also be capable to be extended to also work on phases that does not strictly adhere to the structural principles, like β' , U2 and vacant columns²⁰. Also, if the statistical models can become sophisticated enough, it should also be possible to take precipitates in other alloys into account, like alloys containing²¹ Zn or Ag. How much these methods can potentially be generalized, is still an open question here, and is an exciting topic.

6.6 Optimizations

The current implementation of AutomAl 6000 has many as of yet un-optimized areas, many of which where optimizations are desirable. Especially for large images, some of the GUI functionality can feel a bit unresponsive, making the software feel sluggish. It is perhaps even more important to improve the responsiveness of the GUI, than it is to increase the speed of the algorithms, which already is not too bad on a decent computer²². This is because a slow GUI will often feel more frustrating to a user,

²⁰Like in the particular β'' arrangement, discussed in figure 13b in [5].

²¹Some of these alloying elements can take on different symmetries and structural functions in the atomic structures, but if the advanced species nominal attribute capture all possibilities, they might be separable from the other categories.

²²For an image of approximately 4000 columns (which is the case for the image in figure 3.7b), expect a time of 5-10 minutes for column detection, and 5-ish minutes for column characterization, on a typical office computer. Manual

than the time it takes AutomAl 6000 to run column detection or column characterization, because while the algorithms are running, the user is free to do other things, while that would not be the case while waiting for the response from a button click. Deleting, moving or creating new vertices is particularly slow, due to how these actions are implemented in the background. Improving the responsiveness in these, and similar actions, is likely to improve the user experience of the software.

6.7 Improvements to the GUI

Some work is still needed on the GUI, in particular, many of the elements are yet to be updated to reflect the new functionality of the new data-manager module, as well as the new graph module. Some increased customization capabilities of the graphical elements are also desirable. All though the GUI receives very little attention in this text, due to not being very relevant from the perspective of a thesis in physics or mathematics, it is still an important consideration if the sanity of a potential researcher using AutomAl 6000 extensively, is to be preserved.

6.8 Web-page and bug-tracking

For AutomAl 6000 to become a useful tool to the community, it could do with a more professional distribution practice, than what it has currently. These practices, like a proper issue-tracking practice on GitHub and a professional looking web-page, is a bit outside the scope of this thesis work, but should be considered going forward, on account of the potential longevity of AutomAl 6000 as a useful tool.

revision of column detection result can be expected to take approximately 10 minutes, and manual revision of the column characterization can take anything between 1 minute and 1 hour, depending on the contents of the image.

References

- [1] Stephen J. Pennycook and Peter D. Nellist. *Scanning Transmission Electron Microscopy*. 1st ed. Springer-Verlag GmbH, Apr. 1, 2011. ISBN: 1441971998. DOI: 10.1007/978-1-4419-7200-2.
- [2] Zhang Qing-Hua, Xiao Dong-Dong, and Gu Lin. “Aberration-corrected scanning transmission electron microscopy for complex transition metal oxides.” In: *Chinese Physics B* 25.6 (2016), p. 066803.
- [3] Charles Kittel. *Introduction to solid state physics*. 8th ed. Wiley, 2005. ISBN: 978-0-471-41526-8.
- [4] Sigmund J. Andersen, Calin D. Marioara, Jesper Friis, Sigurd Wenner, and Randi Holmestad. “Precipitates in aluminium alloys”. In: *Advances in Physics: X* 3.1 (Jan. 2018), p. 1479984. DOI: 10.1080/23746149.2018.1479984.
- [5] Takeshi Saito, Eva A. Mørtzell, Sigurd Wenner, Calin D. Marioara, Sigmund J. Andersen, Jesper Friis, Kenji Matsuda, and Randi Holmestad. “Atomic structures of precipitates in Al–Mg–Si alloys with small additions of other elements”. In: *Advanced Engineering Materials* 20.7 (Apr. 2018), p. 1800125. DOI: 10.1002/adem.201800125.
- [6] Sigmund J. Andersen, Calin D. Marioara, Jesper Friis, Ruben Bjørge, Qiang Du, Inga G. Ringdalen, Sigurd Wenner, Eva A. Mørtzell, Randi Holmestad, Takeshi Saito, Jostein Røyset, and Oddvin Reiso. “Directionality and Column Arrangement Principles of Precipitates in Al–Mg–Si (Cu) and Al–Mg–Cu Linked to Line Defect in Al”. In: *Materials Science Forum* 877 (Nov. 2016), pp. 461–470. DOI: 10.4028/www.scientific.net/msf.877.461.
- [7] Gary Chartrand, Linda Lesniak, and Ping Zhang. *Graphs and Digraphs*. 6th ed. CRC Press, 2016. ISBN: 978-1-4987-3576-6.
- [8] Jørgen Bang-Jensen and Gregory Z. Gutin. *Digraphs: Theory, Algorithms and Applications (Springer Monographs in Mathematics)*. 2nd ed. Springer, 2008. ISBN: 978-1-84800-997-4.
- [9] Kenneth Rosen. *Discrete mathematics and its applications*. 6th ed. McGraw-Hill Higher Education, 2007. ISBN: 978-0-07-288008-3.
- [10] Allan Gut. *An Intermediate Course in Probability*. Springer-Verlag GmbH, June 23, 2009. ISBN: 1441901612. URL: https://www.ebook.de/de/product/8237690/allan_gut_an_intermediate_course_in_probability.html.
- [11] Abdelmonem Afifi, Susanne May, and Virginia A. Clark. *Practical multivariate analysis*. Boca Raton: Taylor & Francis, 2012. ISBN: 9781439816806.
- [12] Magnus Nord, Per Erik Vullum, Ian MacLaren, Thomas Tybell, and Randi Holmestad. “Atomap: a new software tool for the automated analysis of atomic resolution images using two-dimensional Gaussian fitting”. In: *Advanced Structural and Chemical Imaging* 3.1 (Feb. 2017). DOI: 10.1186/s40679-017-0042-5.
- [13] *Digital Micrograph*. Version 3.4. Gatan, inc. URL: <https://www.gatan.com/products/tem-analysis/gatan-microscopy-suite-software> (visited on 07/10/2020).
- [14] Pauli Virtanen et al. “SciPy 1.0: fundamental algorithms for scientific computing in Python”. In: *Nature Methods* 17.3 (Feb. 2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.
- [15] Olga Wodo, Srikanta Tirthapura, Sumit Chaudhary, and Baskar Ganapathysubramanian. “A graph-based formulation for computational characterization of bulk heterojunction morphology”. In: *Organic Electronics* 13.6 (June 2012), pp. 1105–1113. DOI: 10.1016/j.orgel.2012.03.007.

- [16] Sai K. Samudrala, Olga Wodo, Santosh K. Suram, Scott R. Broderick, Krishna Rajan, and Baskar Ganapathysubramanian. “A graph-theoretic approach for characterization of precipitates from atom probe tomography data”. In: *Computational Materials Science* 77 (Sept. 2013), pp. 335–342. DOI: 10.1016/j.commatsci.2013.04.038.
- [17] Ethem Alpaydin. *Introduction to machine learning*. Cambridge, Mass: MIT Press, 2004. ISBN: 9780262012119.
- [18] Maxim Ziatdinov, Ondrej Dyck, Artem Maksov, Xufan Li, Xiahan Sang, Kai Xiao, Raymond R. Unocic, Rama Vasudevan, Stephen Jesse, and Sergei V. Kalinin. “Deep Learning of Atomically Resolved Scanning Transmission Electron Microscopy Images: Chemical Identification and Tracking Local Transformations”. In: *ACS Nano* 11.12 (Dec. 2017), pp. 12742–12752. DOI: 10.1021/acsnano.7b07504.
- [19] Jacob Madsen, Pei Liu, Jens Kling, Jakob Birkedal Wagner, Thomas Willum Hansen, Ole Winther, and Jakob Schiøtz. “A Deep Learning Approach to Identify Local Structures in Atomic-Resolution Transmission Electron Microscopy Images”. In: *Advanced Theory and Simulations* 1.8 (July 2018), p. 1800037. DOI: 10.1002/adts.201800037.
- [20] Elisabeth Thronsen, Calin D. Marioara, Jonas K. Sunde, Kazuhiro Minakuchi, Tetsuya Katsumi, Iven Erga, Sigmund J. Andersen, Jesper Friis, Knut Marthinsen, Kenji Matsuda, and Randi Holmestad. “The effect of heavy deformation on the precipitation in an Al-1.3Cu-1.0Mg-0.4Si wt.% alloy”. In: *Materials & Design* 186 (Jan. 2020), p. 108203. DOI: 10.1016/j.matdes.2019.108203.
- [21] Jonas K. Sunde, Calin D. Marioara, Antonius T. J. van Helvoort, and Randi Holmestad. “The evolution of precipitate crystal structures in an Al-Mg-Si(-Cu) alloy studied by a combined HAADF-STEM and SPED approach”. In: *Materials Characterization* 142 (Aug. 2018), pp. 458–469. DOI: 10.1016/j.matchar.2018.05.031.
- [22] Jonas K. Sunde, Calin D. Marioara, and Randi Holmestad. “The effect of low Cu additions on precipitate crystal structures in overaged Al-Mg-Si(-Cu) alloys”. In: *Materials Characterization* 160 (Feb. 2020), p. 110087. DOI: 10.1016/j.matchar.2019.110087.
- [23] Marat Gazizov, Calin Daniel Marioara, Jesper Friis, Sigurd Wenner, Randi Holmestad, and Rustam Kaibyshev. “Precipitation behavior in an Al-Cu-Mg-Si alloy during ageing”. In: *Materials Science and Engineering: A* 767 (Nov. 2019), p. 138369. DOI: 10.1016/j.msea.2019.138369.
- [24] Marat Gazizov, Calin Daniel Marioara, Jesper Friis, Sigurd Wenner, Randi Holmestad, and Rustam Kaibyshev. “Unique hybrid precipitate structures forming in an Al-Cu-Mg-Si alloy”. In: *Journal of Alloys and Compounds* 826 (June 2020), p. 153977. DOI: 10.1016/j.jallcom.2020.153977.
- [25] Adrian Lervik, Calin D. Marioara, Maria Kadanik, John C. Walmsley, Benjamin Milkereit, and Randi Holmestad. “Precipitation in an extruded AA7003 aluminium alloy: Observations of 6xxx-type hardening phases”. In: *Materials & Design* 186 (Jan. 2020), p. 108204. DOI: 10.1016/j.matdes.2019.108204.
- [26] Georg Brandl. *Sphinx*. Version 3. URL: <https://www.sphinx-doc.org/en/master/index.html> (visited on 08/03/2020).
- [27] *Inkscape*. Version 1.0. URL: <https://inkscape.org/> (visited on 07/27/2020).
- [28] *Python*. Version 3.8.5. URL: <https://www.python.org/> (visited on 08/03/2020).

- [29] Stéfan van der Walt, S Chris Colbert, and Gaël Varoquaux. “The NumPy Array: A Structure for Efficient Numerical Computation”. In: *Computing in Science & Engineering* 13.2 (Mar. 2011), pp. 22–30. DOI: 10.1109/mcse.2011.37.
- [30] Travis Oliphant. *Guide to NumPy*. 2nd ed. Austin, Tex: Continuum Press, 2015. ISBN: 9781517300074.
- [31] Riverbank Computing Limited. *PyQt5*. URL: <https://pypi.org/project/PyQt5/> (visited on 08/03/2020).
- [32] John D. Hunter. “Matplotlib: A 2D Graphics Environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/mcse.2007.55.
- [33] Fredrik Lundh and Alex Clarck. *Pillow*. Version 7.2.0. URL: <https://pypi.org/project/Pillow/> (visited on 08/03/2020).

Glossary

- adjacency matrix** A matrix representation of a graph. 10
- adjacent arcs** Two arcs are adjacent if their intersection is non-empty. 9
- adjacent from** "A is adjacent from B" indicates a directed arc extending from B to A. 10, 24, 64
- adjacent to** "A is adjacent to B" indicates a directed arc extending from A to B. 10, 24, 64
- adjacent vertices** Two vertices are adjacent if they are connected by an arc. 9, 65
- advanced species** A "higher resolution" classification system than atomic species. 25, 26, 33, 34, 49, 58
- alpha angle** The angles formed between the vectors from a vertex to its three first vertices of its district. 21, 22, 33
- anti-neighbourhood** The set of all vertices that are in the district, but not in the neighbourhood of a given vertex. 18
- approximate atomic radii** A general sense of atomic radii in pixels, given some image scale. Used by AutomAI 6000 during column detection, and in the gamma normalization calculation. 15, 23
- arc** A tuple of vertices indicating a connection between those vertices in a graph. In an un-directed graph, the vertices are unordered. In a digraph, the vertices are ordered. 8
- arc centered subgraph** A subgraph centered at an arc. 20, 30, 31
- arc permutation** An atomic graph operation that will change the direct successor of an arc. 24
- atomap** Open source column detection software. 58
- atomic graph** A digraph object with custom properties used by AutomAI 6000. 1, 16, 33, 34, 48
- atomic overlay** An HAADF-STEM image where the atomic columns are overlaid with figures that encode the atomic species and z-height of each column. 1, 34
- AutomAI 6000** The name of the software developed for this project. 1, 3, 6, 20, 24, 25, 30, 33, 34, 42, 48, 53, 55–59, 69
- average degree** The average vertex degree in a graph. 9
- binary operation** Graph operation which will produce a graph from two graphs. 9
- cardinality** The number of elements in a set. 7
- catastrophic cancellation** . 12
- class** The class of an arc centered subgraph is determined by the cardinalities of its meshes. 31
- column** A column of atoms seen as a single atom projection in atomic resolution TEM. 4
- column characterization** The process of determining the structural details of a precipitate, and to assign descriptions to repeating structures. 14, 24, 27, 30, 33, 34, 48, 53, 58, 59, 69

column detection The process of detection the 2-dimensional column positions in a HAADF-STEM image. 33, 58, 59, 69

configuration The configuration of an arc centered subgraph is determined by the individual arcs in the subgraph and their symmetry. 31

connected Two vertices are connected if there exists a path between them. 9

connected graph A graph where all vertices are connected. 9

crystal direction Spatial direction vectors in crystals, typically described with so-called Miller indices. 4

crystal lattice The mathematical Bravais lattice, which convoluted with some base, forms the mathematical abstraction of a crystal. 4

deep copy Copy by assigning the creation of a new instance. 15

default model The default model that "ships" with AutomAl 6000. 25, 26, 33, 46, 49

difference The difference between two sets contains all the elements that occur only in the leftmost set. 7

digraph A graph with directed arcs. In some texts this classification includes multigraphs and pseudographs. In this text, a digraph is implicitly a simple digraph.. 10, 17, 64

direct predecessor See adjacent from. 10

direct successor See adjacent to. 10, 24

directed graph See digraph. 10

disjunctive union . 7, 67

district A list of the local vertices to a given vertex ordered by perceived relevance. 17, 53

docstring A string of text in the preamble of a method or class, which which clarifies the input/output and purpose of the method or class. A docstring should comply with a specific style guide. 57

elementary operation A simple graph perturbation, also termed a graph edit. 9, 24

graph A combination of a non-empty set of vertices and a possibly empty set of two-element arc subsets of the vertex set. 8

hybrid precipitate A precipitate with two or more phases present in its structure. 5

in-degree The cardinality of the in-neighbourhood of a given vertex. 10

in-neighbourhood The set of all vertices adjacent to a given vertex. 10

in-semi-partner A vertex is in the in-semi-partner of another vertex, if it is in the in-neighbourhood of that vertex, but not the out-neighbourhood of that vertex. 18

induced subgraph A subgraph build from a subset of a vertex set such that the arc set includes all valid arcs. 9, 20

Inkscape Vector graphics software. 57

intersection The intersection of two sets contains all elements that occur in both sets. 7

interstitial In between. In the context of Al, positions that don't lie on any of the FCC lattice points. 6

inverse arc If a directed arc extends from A to B, the its inverse extends from B to A. If the arc set contains the inverse of an arc, both arcs are symmetric. 10

iterable Any data type that can be iterated over. Examples in python includes lists, sets, dictionaries, tuples and strings. 7

loop An arc connecting a vertex to itself. 10

machine learning Computer base method for training neural networks to solve problems. 55

map The collection of sets that define the graph at a local level for a given vertex. 18

membership The objects in a set are members of that set, and not members of the complement of that set. 7

mesh centered subgraph A subgraph centered at a mesh. 20

Miller indices A system for indexing directions and planes in crystals. 4

multigraph A graph the can contain more than one identical arc. 10

multivariate normal distribution A normal distribution on a multidimensional random variable. 12

neighbour Two vertices are neighbours if they are adjacent, see adjacent vertices. 9

neighbourhood The set of all vertices that are adjacent to or from a given vertex. 9, 17

neural network A graph-like structure used by machine learning and deep learning methods. 55

nominal attribute A categorical data attribute. 11

normal probability plot A plot of the standard scores against the data values. If the data falls along a straight line, it is a positive test for normality in the data, with some caveats. 12, 49

normalized average gamma The average pixel values in a circular area with radius equal to the approximate atomic radii, centered at a column, and after normalization against the Al matrix. 23, 58

normalized peak gamma The brightest pixel value of a given column after normalization against the Al matrix values. 23, 58

numerical attribute A numerical data attribute. 11

numerical wildness . 13, 26

order The order of a graph is equivalent to the cardinality of the vertex set of that graph. 8

out-degree The cardinality of the out-neighbourhood of a given vertex. 10, 17

out-neighbourhood The set of all vertices adjacent from a given vertex. 10, 53

out-semi-partner A vertex is the out-semi-partner of another vertex, if it is in the out-neighbourhood of that vertex, but not the in-neighbourhood of that vertex. 18, 25

overhead parameter A parameter used by AutomAl 6000 during column detection. 15

partner Two vertices are partners if they are both adjacent to each other. 18, 30

pivot vertex The unchanged vertex in an arc permutation operation. 24

projected separation The plane component of the separation, see separation. 17, 33

projected separation matrix A matrix containing all the projected separations in an atomic graph.
27

pseudograph A graph that allow loops, that is, arcs that connect a vertex to itself. 10

search matrix A matrix used b AutomAl 6000 to define the search-space for new columns during column detection. 15

semi-partner Two vertices are semi-partners if they are adjacent by an asymmetric arc, that is, and arc that has no inverse. 18

separation The spatial distance between the center of atoms. 17, 66

set A collection of unique objects of a specific type. 7

set-builder notation A notational method to define sets by some rule on some range.. 7, 9, 17, 53

simple digraph A digraph classification which excludes multigraphs and pseudographs. 10

size The size of a graph is equivalent to the cardinality of the arc set of that graph. 8

smart align A HAADF-STEM image acquisition technique, that matches and overlays information from several images. 28

species dictionary The classification system applied by AutomAl 6000. 26

spectral graph theory The study of the properties of the adjacency matrix representation of graphs.
10

standard score The standard score of a data value, is the mapping of that value in the standardization of its distribution, which is obtained by subtracting the mean and dividing the result with the standard deviation. 12

strong arc preservation Creating the inverse of an un-symmetric arc by increasing the symmetry number of the direct successor of the arc. 25

strong arc termination Removing an arc by, if possible, reducing the symmetry number of the predecessor vertex of the arc. 25

strong operation An atomic graph operation that can permute districts, and also symmetry numbers or atomic species. 24, 25

subgraph A graph where the vertex and arc sets are subsets of another graph. 9

super saturated solute solution A state of an alloy when the alloying elements are homogeneously distributed in the alloy, occupying crystal positions. 5

symmetric difference See disjunctive union. 7

symmetric digraph A digraph that for each arc, also contains its inverse. 11

symmetry The symmetry of a column is the number of opposite plane NNs as seen in projection for a given column in the model description of the structural principles. 6

symmetry number The symmetry number of a vertex determines the size of its out-neighbourhood, that is, its out-degree. 17, 53

theta angle The central angles formed in a vertex centered subgraph. 22

unary operation Graph operation which will produce a graph from a graph. 9

undirected graph A graph with undirected arcs. 10, 48

union The union of two sets contains all the elements that occurs in either set or both. 7, 17

untangling The process of solving an atomic graph into its most symmetric version, given a subgraph operation map. 31, 33, 43

vertex An abstract node acting as the fundamental building block of a graph. 8, 53

vertex centered subgraph A subgraph centered at a vertex. 20

vertex degree The degree of a vertex is equivalent to the cardinality of the open neighbourhood of the vertex. 9

weak arc preservation Creating the inverse of an un-symmetric arc by finding a candidate arc for weak arc termination that is adjacent from the successor of the arc to be preserved. 25

weak arc termination Removing a vertex by permuting the arc to a new successor, if a suitable successor candidate is found. 24

weak operation An atomic graph operation that can permute districts, but not symmetry numbers or atomic species. 24

Z-contrast The number of electrons scattered by the sample, indicated as a pixel intensity in HAADF-STEM images. 3

zeta analysis The election system for determining the zeta values throughout the atomic graph. 30, 38, 53, 58

zeta graph Similar to an atomic graph, except that the definition of the out-neighbourhoods is slightly altered to take zeta values into account. 53, 58

zeta value A Boolean value which represents the plane affiliation of the vertex. Zeta equal to 0, eludes to the $z = 0$ plane, while zeta equal to 1, eludes to the $z = a/2$ plane. 16, 53

A Source code overview

This section provides an overview of AutomAl 6000’s source code, its modules and how they are connected. More in depth and up to date technical documentation can be found at <http://automal.org>.

A.1 Overview

In general, there are four main parts to the software. The modules related to the GUI, a central core which handles the project files and acts like an *Application Programming Interface (API)* to the GUI, the modules that implement atomic graphs and the column characterization methods, and finally peripheral utility modules and other parts that are used by all parts of the software. Figure A.1 shows the import structure of the source code.

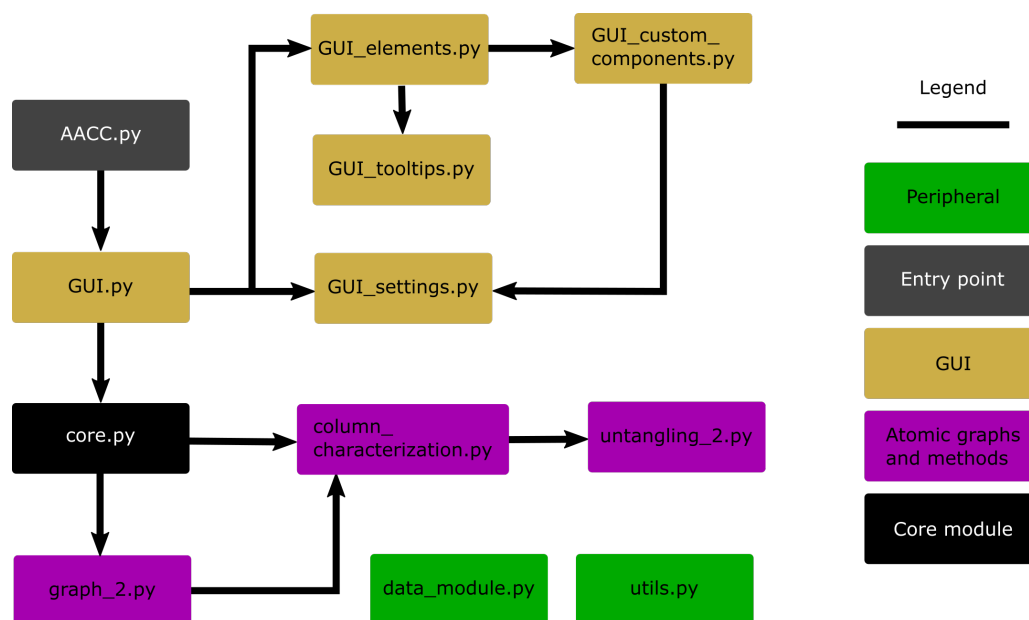


Fig. A.1: An import diagram of AutomAl 6000’s source code. Note that the core functionality can operate without the GUI, which was always a desired design aspect, so that AutomAl 6000’s methods can be imported elsewhere, even if it is not optimally configured for that as of yet. The column detection method is contained in the core module.

A.2 Dependencies

AutomAl 6000 has been mostly tested with python 3.8 [28]. The dependencies of AutomAl 6000 that are not included in the standard python library, are NumPy [29, 30], PyQt5 [31], Matplotlib [32], SciPy [14] and Pillow [33]. AutomAl 6000 also uses a seemingly unsupported DM3 reader script module written by Gregory Jefferis²³. Specific versions are given in the `pipfile` accompanying the source code available from <http://automal.org>.

²³We were unsuccessful at finding a proper reference for this module. The DM3 file format is a protected and closed file format, which might be why this script is not widely supported.

