Universidade do Minho
Escola de Engenharia
Departamento de Informática

André Sá Silva

Efficient computational methods to index crystallographic (S)TEM images and ED patterns

October 2018

Universidade do Minho
Escola de Engenharia
Departamento de Informática

André Sá Silva

Efficient computational methods to index crystallographic (S)TEM images and ED patterns

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
Alberto José Proença
Daniel Grando Stroppa

October 2018

ACKNOWLEDGEMENTS

This dissertation is the culmination of one of my personal goals. However, this was not possible without the help of some special people. They were the ones that motivated me into finishing this work. As such, I would like to acknowledge their importance on this dissertation.

A special thank you to Prof. Alberto Proença and Dr. Daniel Stroppa for the work, patience and comprehension after my ingression on professional work and having less time available to finish this dissertation. Truly their contribution was immeasurable for this work.

A special thanks also for my family which supported me in my decisions and motivated me in the completion of this masters degree.

I would like to thank my coworkers for the words of encouragement and keeping track of the development.

Finaly, I would like to thank the Algoritmi Centre at Universidade do Minho for the research grant and the International Iberian Nanotechnology Laboratory for the access to their installations.

ABSTRACT

Electron Microscopy (EM) of nanomaterials relies on grey-scale images to display the material's atomic arrangement, and a high resolution EM can simultaneously capture multiple atomic structures into a single image. However, the extraction of useful information from these images is still limited to the determination of the material's orientations, an underutilisation of the powerful features of an EM equipment and less productive EM sessions. This is due to the compute-intense tasks that have not been automated yet.

This dissertation aims to significantly reduce the time required to extract useful data from EM images and to remove the user bias when analysing high resolution (S)TEM images, by automating most user routine tasks and integrating them into a software tool, Im2Cr.

The deployed Im2Cr tool aimed to aid an EM user to find the most probable atomic structure orientation of a nanomaterial in a single 2D image from a set of pre-defined materials, with a minimal user interaction.

Im2Cr was designed and built with a simple and intuitive Graphical User Interface (GUI) that runs on a common modern laptop. It takes as input a high resolution (S)TEM image and multiple CIF files with candidate atomic structures to describe the material under observation. After performing the Fourier Transform (FT) on selected Regions Of Interest (ROI) in the image, the tool automatically detects periodic information related to the atom's positions by the brighter spots on the image FT. With a set of geometric computations it tries to match the theoretical values computed with the measured ones by assigning a custom made merit index. This quantitative evaluation avoids possible user bias and/or errors on image characterisation. Im2Cr outputs at the end a report with the best matching crystallographic structure, its orientation and the indexation table.

This tool was successfully tested for robustness and execution efficiency in a wide range of high resolution (S)TEM images from crystalline nanomaterials, with domain size ranging from 4 to 100 nm. The autonomous indexation with preset parameters has a very high success rate and runs in a small fraction of typical (S)TEM images acquisition time by taking advantage of the inherent hardware parallelism. Alternatively, the user can change some relevant parameters related to the ROI selection on the (S)TEM image and on the FT peaks detection.

Im2Cr promising results point to the possibility of real-time image analysis with reduced user interaction, allowing for an increased (S)TEM characterisation yield and also enabling the interpretation of complex images, such as those from nanocrystalline materials imaged in high-order zone axis orientations.

RESUMO

A microscopia eletrónica (EM) de nanomateriais usa imagens em escala de cinza para representar a estrutura atómica de um material, sendo a EM de alta resolução capaz de capturar simultaneamente múltiplas estruturas atómicas numa única imagem. No entanto, a extração de informação útil destas imagens ainda é limitada pela determinação das orientações do material representado, o que resulta numa subutilização de equipamentos de EM e em sessões menos produtivas. A grande razão para esta limitação deve-se à atual baixa automação de tarefas computacionalmente intensivas necessárias para a caracterização do material.

Esta dissertação tem como objetivo reduzir significativamente o tempo necessário para extrair informação das imagens EM, removendo da equação uma possível análise tendenciosa inconsciente do utilizador através da automação deste processo numa nova ferramenta, Im2Cr.

O Im2Cr tem como objetivo ajudar o utilizador a encontrar a orientação da estrutura atómica mais provável de determinado nano material a partir de um conjunto de materiais pré-definidos e uma única imagem 2D, com o mínimo possível de interação do utilizador.

O Im2Cr foi desenhado e construído para fazer uso de uma simples e intuitiva interface gráfica (GUI) capaz de ser executada num normal computador pessoal. Esta aplicação recebe como dados de entrada uma imagem (S)TEM de alta resolução e vários ficheiros CIF com as estruturas atómicas candidatas para descrever o material observado. Após aplicação da Transformada de Fourier (FT) na região de interesse (ROI) selecionada na imagem, a ferramenta é capaz de detetar automaticamente informação periódicas relativa às posições dos átomos através dos pontos mais claros na imagem FT. Com base num conjunto de cálculos geométricos, a aplicação tenta combinar os valores teóricos calculados com os valores medidos, avaliando assim as correlações com base num recém-criado índice de mérito. Esta avaliação quantitativa evita possíveis influências do utilizador e/ou erros de cálculo na caracterização da imagem. No final, o Im2Cr exporta um relatório com a melhor estrutura cristalográfica encontrada, a sua orientação e a respetiva tabela de indexação.

Esta ferramenta foi submetida a testes de robustez e eficiência de execução com base numa ampla variedade de imagens (S)TEM de alta resolução de nanomateriais cristalinos, cujo tamanho variava entre 4 a 100 nm. A indexação autónoma com uso de parâmetros predefinidos tem uma taxa de sucesso significativa e é executada numa fração do tempo quando comparada com o tempo típico de captura de imagens (S)TEM, fazendo uso do paralelismo de hardware existente. Alternativamente, o utilizador tem o poder de poder alterar alguns parâmetros relevantes relacionados à seleção da ROI na imagem e à deteção de picos na FT.

Os resultados obtidos apontam para a possibilidade da análise de imagens em tempo real com uma reduzida interação do utilizador, permitindo assim um aumento do desempenho na caracterização de imagens (S)TEM e possibilitando ainda a interpretação de imagens mais complexas, nomeadamente imagens de materiais nanocristalinos com orientações menos convencionais.

CONTENTS

1

INTRODUCTION

1.1 CONTEXT

The study of the atomic structures in nanotechnology has a significant relevance at the present time, since it is an area that reveals the potential application of different materials to different areas of research.

The potential applications of such materials are revealed by its properties, which depends on their structure at the atomic level. At the scale below 100 nanometres, nanotechnology research aims to explore materials properties in order to better suit real world application, for example, transistors for the semiconductor industry or catalysts for chemical industry.

To improve nanostructured devices performance, it is necessary to understand and optimise their atomic arrangement. Materials characterisation techniques with high spatial resolution are fundamental for this task.

In nanotechnology research conventional optical microscopes do not have the required resolution to be able to separate individual columns of atoms. An Electron Microscope overcomes this problem by using a beam of electrons instead of a light source, being able to achieve resolutions bellow ångströms.

Electron Microscopy (EM) uses one of two main methods to capture atomic structure images: Transmission Electron Microscopy (TEM) or Scanning Transmission Electron Microscopy (STEM).

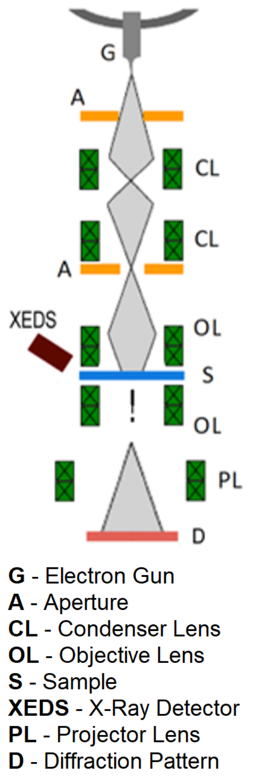


Figure 1.: TEM microscope schematic

TEM is a versatile tool that can be used to characterise materials. The use of high-energy electrons allows an improved resolution with respect to optical microscopy - modern (S)TEM equipments can reach resolution down to 50 picometers.

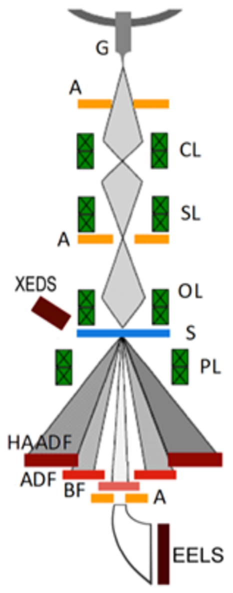
In addition to atomic resolution imaging, (S)TEM is able to provide information about the atomic arrangement through electron diffraction and chemical information species quantification through spectroscopy.

TEM is based on a broad electron beam that illuminates a large field of view (up to several microns) of a thin sample. At lower magnifications, images are formed similarly to an optical microscope in transmission mode.

To retrieve the atomic structure from periodic (crystalline) samples, TEM can be used to directly generate Electron Diffraction (ED) patterns, or to promote the interference of diffracted electrons to form atomic resolution images. Both approaches generate images containing information on the periodicity of the sample structure.

STEM is based on a small (0.1 nm or smaller) electron beam that is scanned over an area of interest of a thin sample. The transmitted electrons can be detected by a set of detectors in different annular ranges. The most widely used is the High Angle Annular Dark Field (HAADF) region, which comprise electrons scattered at high angles (from 80 milliradians) and can be used directly to form atomic resolution images.

Both TEM and STEM approaches lead to high resolution images containing information on the sample's atomic structure. However, a demanding procedure of image analysis is required to their interpretation.



G - Electron Gun
A - Aperture
CL - Condenser Lens
SL - Scanning Lens
OL - Objective Lens
S - Sample
XEDS - X-Ray Detector
PL - Projector Lens
BF - Bright Field Detector
ADF - Annular Dark Field Detector
HAADF - High Angle ADF Detector
EELS - Electron Energy Loss Spectrometer

Figure 2.: STEM microscope schematic

1.2 MOTIVATION AND GOALS

The usage of EM technology to characterise materials is well established on today's industry as a mean to find the best suited materials for a specific application. As such, researchers are constantly experimenting with new materials combinations to study their properties and potential applications.

An important step for this study is to know which material is being displayed as well as its orientation. Using this information, it is possible to perform adjustments to achieve a desired result.

One example of application of image characterisation is the reconstruction of a 3D model of the material being studied. In this process it is required to know which material, orientation and dimensions prior to try to simulate a 3D model.

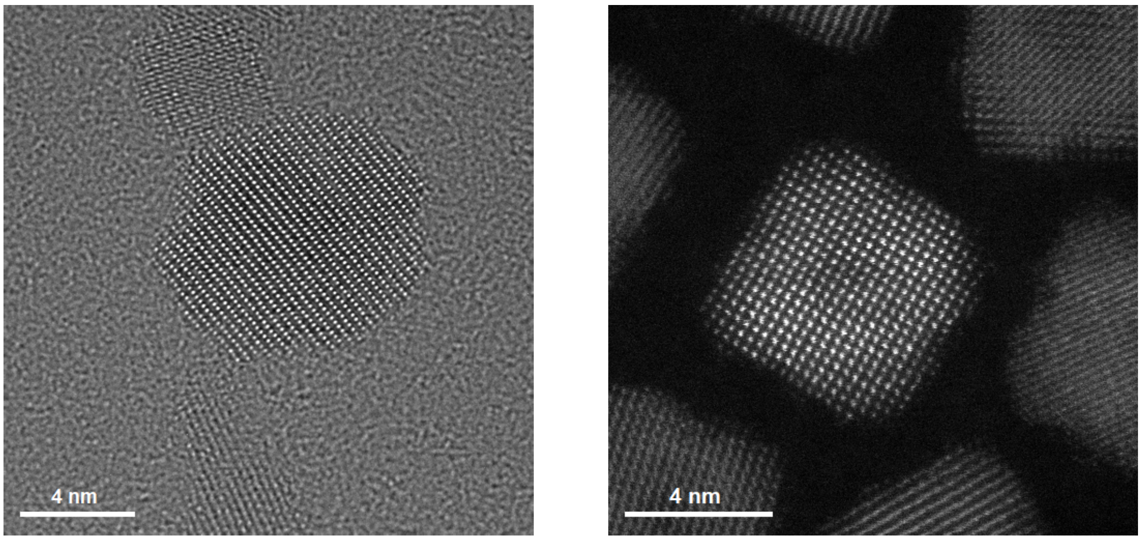


Figure 3.: Example of TEM (a) and STEM (b) images of CeO_2

The validation of the three-dimensional analysis using two-dimensional images is already proven through different methods (Stroppa et al., 2013) (Jia et al., 2014), having all in common the process of validation by comparing the experimental image with the image resulting from the application of the simulation process to the atomic structure model expected.

However, the analysis of the experimental image still requires human intervention in the delimitation of Regions Of Interest (ROI), as well as expertise in the determination of the atomic structure model and its orientation. These requirements result in the under-utilisation of TEM microscopy technology due to, in large part, the time required to characterise each acquired image.

This dissertation aims to develop a new autonomous tool, Im2Cr (stands as image to crystal). This application should be used to extract crystalline structure information from (S)TEM images in a process known as indexing. The expected output should be a model of the observed material in the correct orientation.

With regards to user requirements, the tool should be easy to use by having a robust and user-friendly Graphical User Interface (GUI) and should be validated with several real TEM, STEM and ED images. The user experience should also be taken into consideration by being responsive and performing the task quickly. Also, the tool should be usable with current laptops/desktops.

1.3 CONTRIBUTION

To address the previously mentioned problem, the introduction of a semi-autonomous tool which require little user interaction and can perform multiple image characterisation should greatly reduce the time needed for the analysis of experimental images. Also, the use of computational power to perform extensive image analysis should enable the characterisation of images which, in current manual process, would not be used due to, for instance, a high order zone axis in which the atomic structure is represented in the image.

To achieve this, the developed software should receive as user inputs the images to be indexed and the Crystallographic Information File (CIF) regarding to the expected unit cells to be in the image and output the best suited material found and its orientation. To be less subjective on the classification of each material, a merit index is applied, which in turn allows less experienced users to use the tool and understand the results displayed.

Regarding the computational performance, the developed application should make use of current modern processors features, namely the hardware support for vectorization and multiple threads at each core, the multitude of the available cores, and a careful balance of the available memory hierarchy.

To minimise the computational penalty of creating and deleting threads, these should not be larger than the number of allocated cores: the developed software should reuse threads when possible, using a task pool, which increases the overall efficiency and reduces the time taken to perform batch indexation.

1.4 DISSERTATION STRUCTURE

This dissertation addresses the current state-of-the-art on (S)TEM images indexation, from the initial developments to the most up-to-date studies and applications in the field.

Next chapter describes the developed application with details on its development, faced challenges and how they were overcome. For an easier readability about the development process, this chapter is approached with a top-down perspective: it describes the available user interface and on the following sections a more detailed description of the inner processes and algorithms.

To demonstrate the contributions made possible with this application, the following chapter benchmarks the application and display its results, addressing the application thread utilisation and the used optimisation, as well as a comparison with a non-optimised version of the application.

The final chapter draws conclusions from a comparison between the application expectations and the work outcomes. This chapter also addresses the application potential by suggesting further developments from this dissertation work.

2

DESIGNING A 3D MODEL FROM AN (S)TEM IMAGE

The process of determination of zone axis and upward vector is a well-known and essential in the analysis of images of electron microscopy. It is from this indexing procedure that it is possible to determine the orientation of the unit cell and, based on this information, to simulate the image obtained from theoretical information about the observed crystal.

The automation of this process has been developed in recent years with resort to different approaches, each having their respective advantages and disadvantages. Likewise, technological evolution at the computational level has seen significant advances that contribute to a faster and easier indexation process.

In this sense, the present chapter will address the existing algorithms and implementations in the field of electron microscopy as well as the main evolutions in computational technology.

2.1 RELATED WORK

The analysis of experimental images is an essential procedure in the crystallographic characterisation in electron microscopy. One of the key steps in this process is to identify the observed crystal and determine its orientation, process also known as indexing.

The theoretical steps of this process are widely known (Bunge, 1982) and are often performed manually. Thus, the indexation of crystals can be considered a time-consuming and error-prone process, although not necessarily complex.

In this sense, efforts have been made to automate this process in a robust and reliable way. One of the first developments of automatic tools is described in a paper by Stefan Zaefferer (Zaefferer, 2000) where two different approaches of indexing the orientation of the observed material are presented.

One of the indexing methods described makes use of Kikuchi's lines displayed in the diffraction pattern image. A very simplified description of the Kikuchi lines is that they are characterised by pixel lines observable in captured images. These lines can then be mapped against a line map of the respective crystal where the goal is to determine the interception between the lines displayed and to which planes these interceptions correspond.

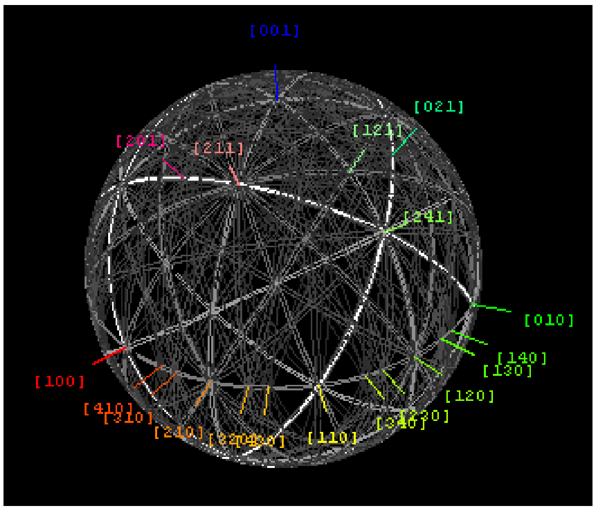


Figure 4.: Kikuchi lines, from Wikipedia

In the second method, it is suggested the indexation from electron diffraction patterns. The diffraction patterns are characterised by regions of higher intensity present in the image and are directly related to the arrangement of the atoms in the sample, since they are the ones that cause the diffraction of the electrons from the beam.

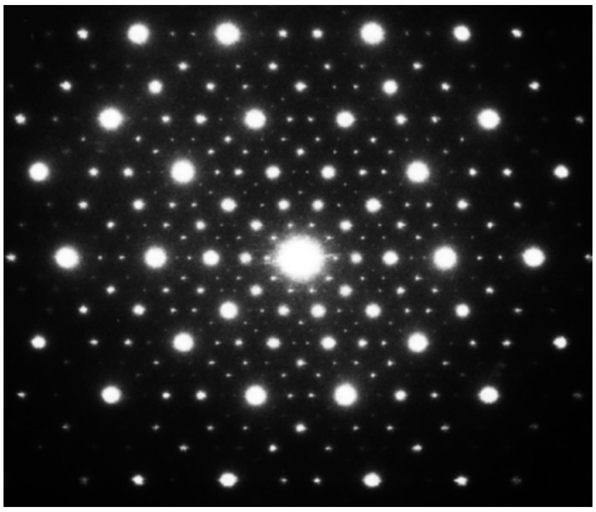


Figure 5.: Example of an electron diffraction image, from Wikipedia

In this study, it was found that indexing through the Kikuchi lines is more sensitive to deformations in the sample, when compared to the diffraction of electrons; but on both it is possible to index the observed crystal.

The automation of the electron diffraction indexation process is described based on four main phases:

- obtaining the image and correctly identifying the coordinates of the spots present;

- determination of the angles and magnitudes of the vectors of the diffraction pattern;
- calculation of the theoretical values for each plane *hkl* and a comparison with the experimental values, assigning the theoretical value according to the tolerance allowed;
- confirmation of correct indexation using the simulation of the crystal structure in the calculated orientation.

To increase indexation accuracy, this paper mentions the use of automation in image processing, to determine the coordinates of the centre of each spot. For this purpose, it is used the centre of gravity of the sum of the pixel intensities in each spot.

As limitations, it could be said that the sample used needs to have some thickness in order to produce the Kikuchi lines. This requirement excludes the feasibility of this process for nanocrystals indexation.

Regarding automatic image analysis, in 2005 a paper related to a new methodology using diffraction patterns was released (E. and Laurent, 2005).

In this paper it is mentioned that the use of a camera capable of registering 8 bits per pixel is enough to perform a good indexation while keeping low image acquisition times.

In this study it is also verified that this level of scanning can be used in images down to a minimum resolution of 128x128.

Regarding the indexation process, a new approach is suggested using pre-calculated and stored templates. A calculation is then made to correlate each template with the obtained experimental image. Regarding the size of the template database, the author of this study considers that about two thousand templates will be enough to index images with a margin of error less than 1 degree.

About the indexation process through image correlation, it is mentioned that the simple use of the correction index is not enough to determine the orientation of the sample. To demonstrate this scenario, a case is exposed where two templates have a very close correlation index, generating ambiguity.

To solve this problem, the author suggests a reliability index, which considers the two highest correlation indices. If the value of both is very close, the total value applying the suggested formula is close to 0. On the other hand, if the values have a significant difference, the result will be close to 100. Using the formula, the study suggests that indexes with values greater than 15 are safe. Likewise, values over 40 are excellent.

With this study it was concluded that the use of templates is useful to reduce the computation time required indexing experimental images.

As a limitation of template matching algorithms, it can be said that the input images must be diffraction pattern images and its indexation process rely on the comparison with all computed patterns, which translates into a considerable amount of useless computations.

On the other hand, this process allows great flexibility by allowing image indexation on images with less than ideal zone axis.

In the following year a criticism was made (Morawiec and Bouzy, 2006) regarding the use of the template-based method, in the sense that it presents ambiguity in the 180 degrees indexation. Additionally, it is suggested that simple voltage variation would eliminate such ambiguity. This paper concludes that the automation of orientation determination based on templates should act as a complement in the analysis of samples, not removing however the merit in the indexation of samples with deformations.

In response to the problem of ambiguity, the author of the 2005's paper wrote in the same publication (E. and Laurent, 2006) to note that this is a more technical than theoretical problem in the sense that this ambiguity can be removed in three distinct ways: (i) the use of more distant spots, (ii) the use of beam stoppers to avoid saturation of the detector and allowing the use of spots with more precise angles, and (iii) the acquisition and indexation of a second image taking into account the changed angle, thus eliminating ambiguity.

In a more recent paper (Meng and Zuo, 2017), one more study was made about the algorithms at the time, where two alternative implementations related to automatic indexation were proposed: (i) an improvement of the algorithm of indexation by template matching, or (ii) a pattern matching algorithm for 2D coordinates, based on the comparison of triangles formed by each set of 3 coordinates.

In the first implementation the improvement takes into consideration the information regarding the angles and distances of each spot, both in the experimental image and in the simulated template. After this process, a 1D Normalised Cross-Correlation (NCC) calculation is performed between the values obtained on both the experimental image and the created simulated patterns. Based on this comparison, a number of simulations are then chosen to be compared to the experimental image using 2D NCC (similar to the original template matching algorithm) using a formula presented in the paper.

In the second implementation, based on a 1986 study for the area of astrology (Groth, 1986), the algorithm calculates the comparison of the triangles between the original image and the simulated image, based on "polls" that represent the possibility of a certain spot being common to both images. The author suggests the use of this algorithm applied to the coordinates of the spots in electron diffraction images.

Regarding the first approach, it is indicated that the improved algorithm of template matching has some limitations, namely the magnification used must be known. The deviation relative to the zone axis can also have an impact on the indexation result. On the other hand, in Groth's triangle-based indexation it is possible to index images with some deviation from the zone axis, but it is also dependent on the good delimitation of the spots in the diffraction pattern.

In terms of execution times, the author reports approximate times of 6 seconds for the improved template-matching implementation in a 256x256 image. For the Groth's triangle implementation it takes about 10 times longer.

Despite this significant increase in execution time, this paper concludes that the use of Groth's triangles is recommended when reliability is desired. On the other hand, the first approach is suggested when the execution times are important, and the diffraction patterns spots are well defined.

An alternative MATLAB-based implementation to this dissertation was published in 2015 (Klinger and Jäger, 2015). This implementation is composed of three individual but complementary applications: (i) diffractGUI, used to index electron diffraction, (ii) ringGUI, to index circular electron diffraction and (iii) cellViewer, which displays a 3D model of a crystal and allows the manipulation and calculation of orientations.

In the diffractGUI application the automatic calculation of the centre of the diffraction patterns can be performed; however, it has a significant impact in terms of performance and execution times. After determining the centre, the indexation is performed using theoretical values of the observed crystal. In this indexation process an automatic routine searches for false positives and false negatives. To achieve this functionality, the RANSAC algorithm was implemented in the application, which, from a number of detected spots, tries to determine two primitive vectors that allow to build the observed lattice. Based on the comparison between the constructed lattice and the experimental image the valid spots are subsequently adjusted.

Similarly, based on the implementation of the RANSAC algorithm, the author states that it is possible to process two or three overlapping diffraction patterns. To achieve this the application executes the algorithm and determines a first lattice. Afterwards the algorithm is executed again, ignoring the spots in the previous lattice.

Regarding the application of circular electron diffraction, ringGUI, the author of the paper states that the application is capable of processing images whose circular shape is not fully formed, as long as the image has not been distorted. In this application it is also possible to use images that contains or not a beam stopper. If it is present, the application interpolates the missing values in order to allow the determination of the centre of the diffraction. As a result, the application shows the reconstructed pattern (in the case of the experiment image containing the beam stopper) and the plane corresponding to each circular diffraction.

Finally, the 3D visualisation application of crystal forms, cellViewer, allows the user to visualise the result of the indexation computed in the previous applications, as well as to display a theoretical simulation of the lattice produced by a given orientation.

With relation to performance, according to the paper, a benchmark was set comparing the time taken by using the application and the time taken by performing the indexation process manually by two experienced analysts. For this test the analysts could use computer tools as long as they did not perform parts of the indexing process automatically. As for the application, it was run on a Lenovo ThinkPad E540 laptop with a 2.3GHz Core i7 processor in a single core. In the benchmark result, the application ran between 12 and 22 seconds for four images, while the two analysts recorded times between 390 and 980 seconds and 960 and 2700 seconds, respectively.

Based on these results, the author suggests that the application allows speedups between 30 and 60 times compared to manual indexing. As a conclusion, the paper highlights the fact that the application allows significant time savings using the application, while recognising that there will always exist difficult cases were the automation shows its limitations.

2.2 END-USER REQUIREMENTS

With the end user requirements, the optimal outcome of this dissertation work would be the deployment of an autonomous tool that, from an input image, could detect the displayed ROIs, select the best suited CIF files and perform the indexation process for each ROI. The software should output the best matching crystallographic structure, its orientation and the indexation table. These steps should be fast enough to be performed during a experimental EM session, right after an image capture, or, in the optimal case, using a video feed as input.

Taking into consideration current image acquisition times, the time taken to capture a STEM image is in the range of 2 to 20 seconds, while TEM images takes around 0.1 to 1 seconds. If we look at the state of the art related to image capture of TEM, there is equipment available that allows video up to 1000 Frames Per Second (FPS).

For system requirements, the software should be able to run on a common modern laptop for single image indexation or a more robust computer for the use case of video as input.

Regarding the Operating System (OS) to be supported, it should be possible to run the software on all main OSs: Linux, Mac OS and Windows, being the latter two the more suitable to be run on the end user laptop and the first one targeted to a more robust computer, mainly a server.

2.3 CHALLENGES

The previous section displayed all the desirable requirements. However, some of those requirements are still beyond current computer capabilities, being these currently minimised with the aid of parallel execution.

As such, there are a number of limitations that prevent all above-mentioned user requirements to be fulfilled. Following are some examples of these limitations that were identified or found during the software development.

2.3.1 *Autonomous ROI selection*

The end user requirements stated that the tool should perform image indexation without user intervention to select the ROIs on the image. Preliminary work was done in order to assess this requirement, which showed some promising results.

As one can see in both images 6a and 6b, the ROI detection displayed in image 6b is a good selection from the input image 6a.

To achieve this result, the input image had its rows and columns scanned and a statistical variation was performed on them. This computation showed that the regions with greater variation were usually where the ROI was located, since the atom's columns are usually represented through a significant pixel contrast when compared with its neighbour pixels. The algorithm then performed a best fit polynomial function of order 5 to the variation results and registered the polynomial maximum and minimum values of the polynomial function. If the difference between these two values was significant, then a ”water line” was set and only the pixel coordinates which had polynomial values above the water line were accepted, being the coordinates below this threshold discarded. Performing this process to the image rows and columns lead to the ROI selection in image 6b.

The algorithm results displayed could lead to believe that autonomous ROI detection should be possible without human intervention. This interpretation was later discovered as deceiving, since this algorithm has an intrinsic limitation to find a single ROI on each input image.

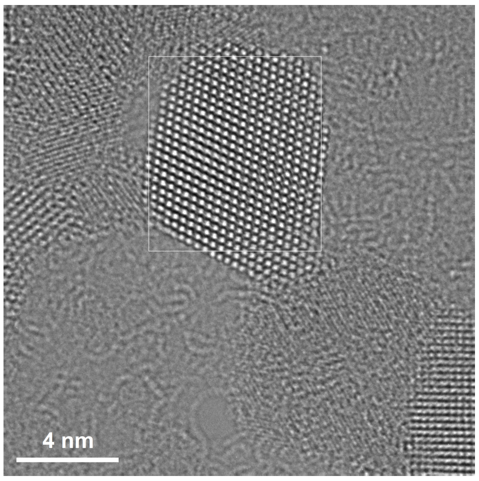
Being EM currently focused on increasing image resolution, the future perspective on image indexation is to use higher resolution images displaying more than one ROI. As such, autonomous multiple ROI detection is currently a challenge to be addressed on future work, being the solution in the meantime to have the user selecting the ROI area.

2.3.2 *Autonomous CIF selection*

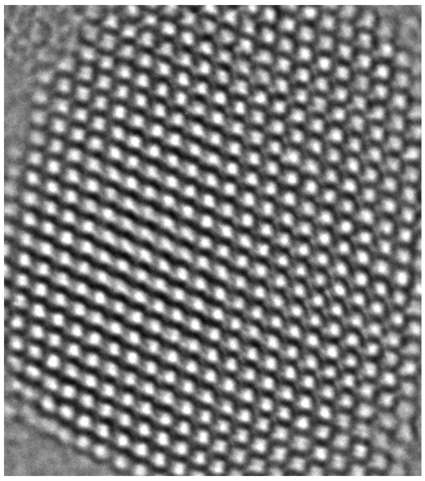
Another desirable software functionality was to automatically select the best suited CIF files in order to perform image indexation.

The main challenge on this functionality is the various similarities present on different unit cells. As an example, one cubic unit cell with its structure displayed on the experimental image could be indexed with another unit cell with double the lateral dimension and some atoms at the middle of each edge. Both CIFs would have the same angles and spacings.

This still is a challenge on its own, being used special tools, namely Energy-Dispersive X-ray (EDX) spectroscopy, to find which elements are present on the sample.

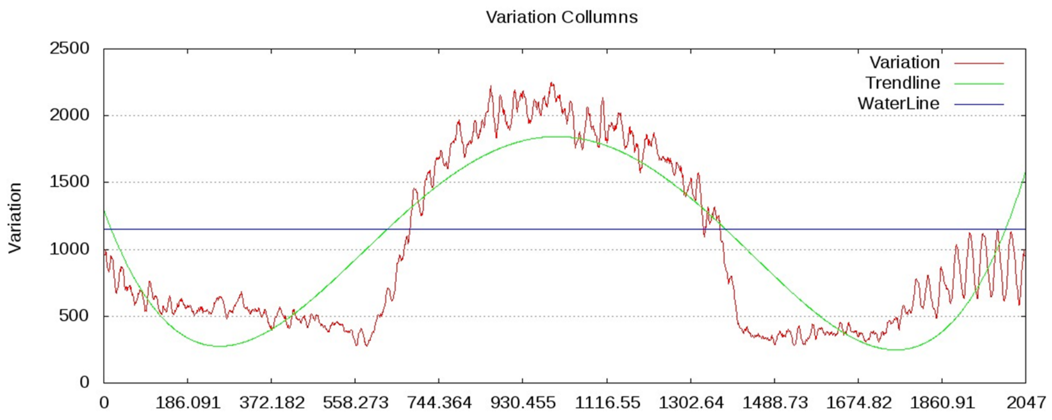


(a) Detected ROI selection

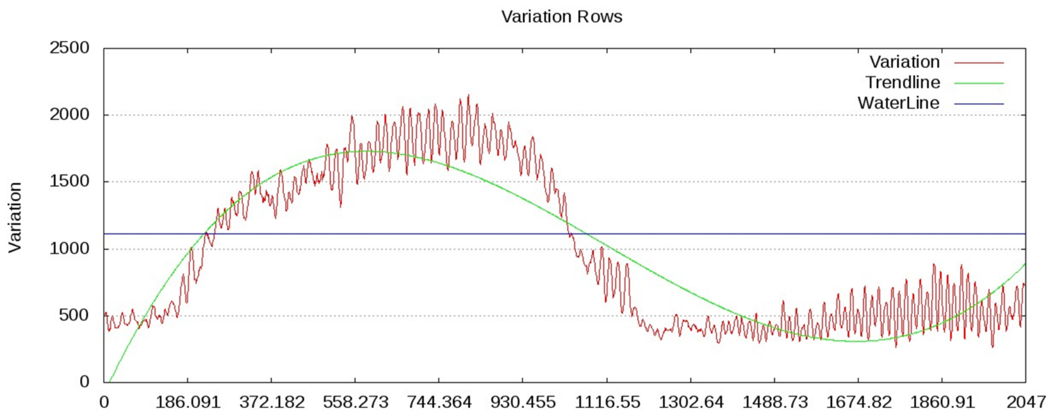


(b) ROI selection

Figure 6.: CeO2 0.009nm with autonomous ROI selection



(a) Variation along image width (left-right)



(b) Variation along image height (top-down)

Figure 7.: CeO2 0.009nm with autonomous ROI selection

The optimal solution would be to perform indexation in parallel, having multiple CIFs being indexed at the same time. This added to the possibility of excluding CIFs based on image information taken after some image segmentation, could make the process more efficient. However, given the number of CIFs available over 400 000 (Crystallography Open Database) it would require a machine with many processor units.

To address this problem, the current software implementation requires the CIF files to be provided by the user, who have some knowledge about the material and should reduce the false positive results.

2.3.3 *Graphical User Interface*

Taking into consideration the previously mentioned challenge of ROI selection, there is a need for user interaction with the software. To solve this problem, the more intuitive solution would be to implement a Graphical User Interface (GUI) where the user can see the image and select on it the desired ROI.

The development of a GUI in itself is not a challenge, but the requirement of a portable GUI implementation added to the time required to learn the Application Programming Interface (API) and implement the code for all the user controls, could restrict the time available to implement useful functionalities and code optimisations.

Also, the use of a GUI should make the greatest bottleneck on the application performance, since user interaction should be measured in the order of the seconds and is hardly optimisable, while the code could be optimised and should execute in the order of milliseconds.

So, to reduce the time taken by user interaction, the challenge is to develop a simple interface that requires the minimum user inputs possible, while being flexible allowing the configuration of the indexation process for more advanced users.

2.3.4 *Parsing of proprietary image formats*

The Im2Cr software should be able to read images in the most popular formats used by EM researchers. Some of these formats are open and in most cases there is already a library implemented to read these formats. A practical example of one of these formats used by the Im2Cr is the TIF or TIFF format, which is already supported by the OpenCV library used in the application.

However, there are many proprietary image formats, whose specifications are not disclosed, being only available information compiled by the research community. Some of such formats used by the Im2Cr tool are the TIA's SER format and the GATAN's DM3 format. Being proprietary formats, its use is confined to the official tools or open software implemented in some other software language without API documentation. As a result, a new parser should be implemented in order to be able to read these formats.

2.3.5 *3D model display*

Since the indexation process aims to find the crystallographic structure orientation defined by its Zone Axis (ZA) and Upward Vector (UV), a 3D representation of the indexed CIF in the correct orientation is required.

In the indexation process the ZA is set as axis coefficients that creates the vector to which the material is being observed while the UV is the vector perpendicular to ZA resulting from a computation using each two spot indexed planes. In a 3D model this could be used as the camera's direction and the up vector respectively.

To satisfy this requirement, an interactive 3D scene should be software rendered. This inclusion implies the implementation of a set of functionalities such as user mouse interaction with the model, scene lighting, atomic representation, atoms colours, etc. Similar to the GUI challenge, even if the implementation of a model render is not complex by its own, the implementation process requires a great time allocation.

2.3.6 *Libraries and its licences*

To minimise the development time, the use of currently available software is encouraged. After selecting an implementation language, if any functionality or component of a functionality was already available through a library, its use should be taken into consideration. This however does not mean that any library should be used. Its API, complexity, performance and the license used should also be taken into consideration.

The API documentation and integration complexity have a significant impact in the application development time, so the choice to use any library should measure its future cost in the development phase and afterwards support.

The library license used is also a factor, since the final destination for the developed software is still unknown. As such, the libraries used should have licenses commercial free when possible or, at least, not be closed for public use.

3

IM2CR: A TOOL TO INDEX CRYSTALLOGRAPHIC IMAGES

Given the user requirements, the objective of this dissertation is to apply current knowledge and develop an integrated application for CIF indexation. The main goal to achieve in this application is to prove its usefulness by reducing the time taken to perform this task while maintaining or improving the results quality. The name given to this application was Im2Cr.

This chapter presents a top-down view of the of the tool, beginning with the GUI from the user perspective and detailing some of the main algorithms used on the following sections.

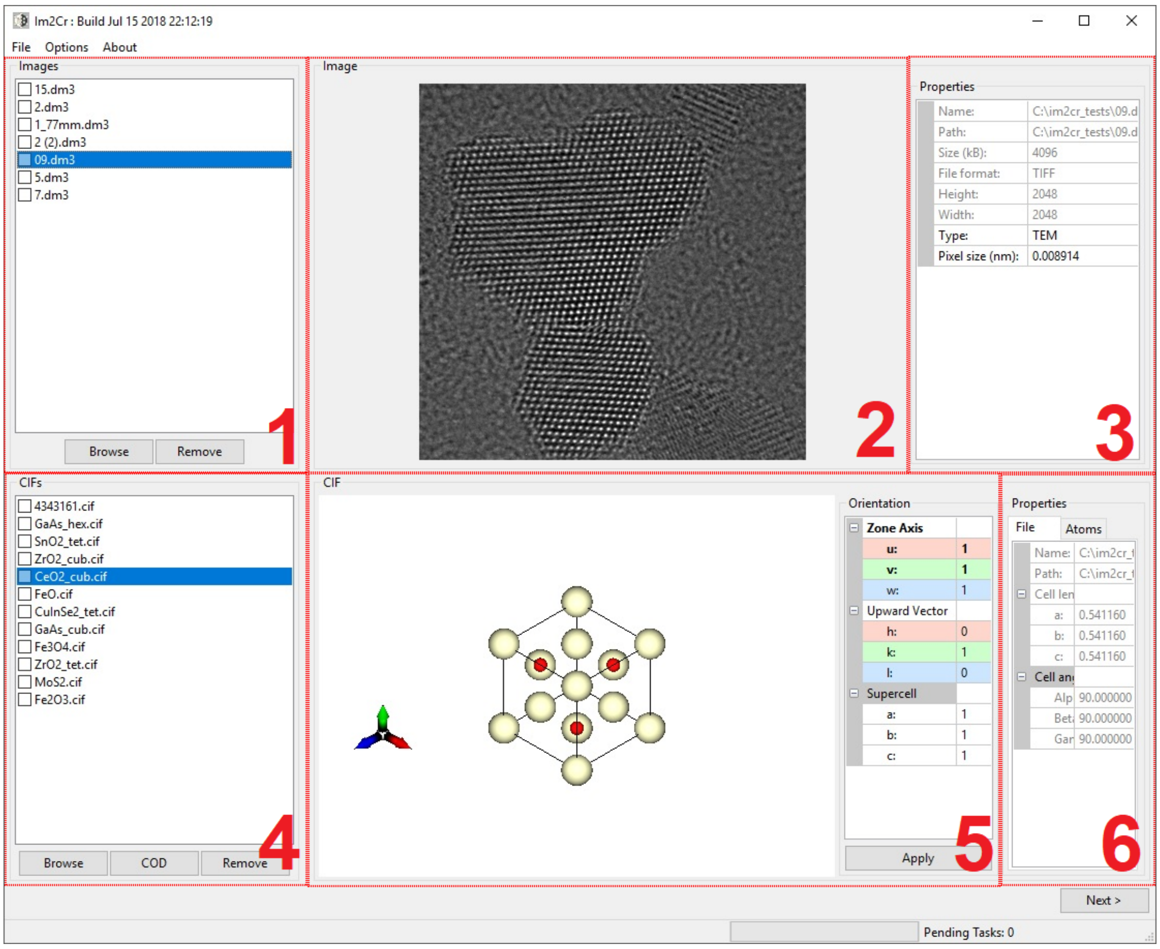
3.1 USER GRAPHIC INTERFACE

Being a tool aimed to be used by researchers of EM, its use should be kept simple and intuitive by not requiring specialised knowledge in order to use the tool. The objective is to get the users to use the tool as fast as possible and not deviate their focus to how to use the tool. With this goal in mind, the use of a terminal based tool which would accept user commands as input and display or output files as result would be in conflict with this requirement. Also, the requirement to select a ROI with image pixels coordinates is easier to do with a mouse over a displaying image then to select coordinates from a text box. For these reasons, the Im2Cr is a cross-platform GUI based tool which aims to require low user input by setting a number of presets found to be appropriate from a set of test images.

The GUI of Im2Cr was developed to display the three main phases: (i) input of images and CIFs, (ii) ROI selection and indexing configuration and (iii) display index results.

3.1.1 *User inputs*

Once the user initialises the application, the file inputs screen is displayed, as shown on image 8. This window may contain six distinct areas:



Images: 15.dm3, 2.dm3, 1_77mm.dm3, 2 (2).dm3, 09.dm3, 5.dm3, 7.dm3

Name:	C:\im2cr_tests\09.d
Path:	C:\im2cr_tests\09.d
Size (kB):	4096
File format:	TIFF
Height:	2048
Width:	2048
Type:	TEM
Pixel size (nm):	0.008914

CIFs: 4343161.cif, GaAs_hex.cif, SnO2_tet.cif, ZrO2_cub.cif, CeO2_cub.cif, FeO.cif, CuInSe2_tet.cif, GaAs_cub.cif, Fe3O4.cif, ZrO2_tet.cif, MoS2.cif, Fe2O3.cif

Zone Axis	
u:	1
v:	1
w:	1
Upward Vector	
h:	0
k:	1
l:	0
Supercell	
a:	1
b:	1
c:	1

Name:	C:\im2cr_
Path:	C:\im2cr_
Cell len	
a:	0.541160
b:	0.541160
c:	0.541160
Cell an	
Alp	90.000000
Bet	90.000000
Gar	90.000000

Figure 8.: Image and CIFs inputs

1. List of input images

   On the top left of the screen, there is a list of currently loaded images. The input images may be in one of three different formats: TIF(F), SER or DM3. This list box allows the user to add or remove one or more images at a time.

2. Preview of the selected image.

   At the top on the middle is the image preview. When the user selects the image on the list, a preview is shown on this area.

3. Display of image properties.

   In this area, some fields are not allowed to be modified. These fields are mainly information about file name, format and image dimensions. However, there are some properties enabled to be modified by the user. These fields are used to inform the software that the selected image is a TEM, STEM or ED image and its pixel size in nanometres per pixel.

4. List of input CIF files.

   Similar to the image selection box, the bottom left contains the list of currently loaded CIFs. This list box also allows the addition or removal of multiple items at the same time.

   There is also a special function to import CIF files directly from an online database. Choosing the COD button the tool opens a window as displayed in figure 9. Here the user can either enter the chemical compounds manually or click on them from the table. After searching and receiving the results the user can then choose which CIF file to download.

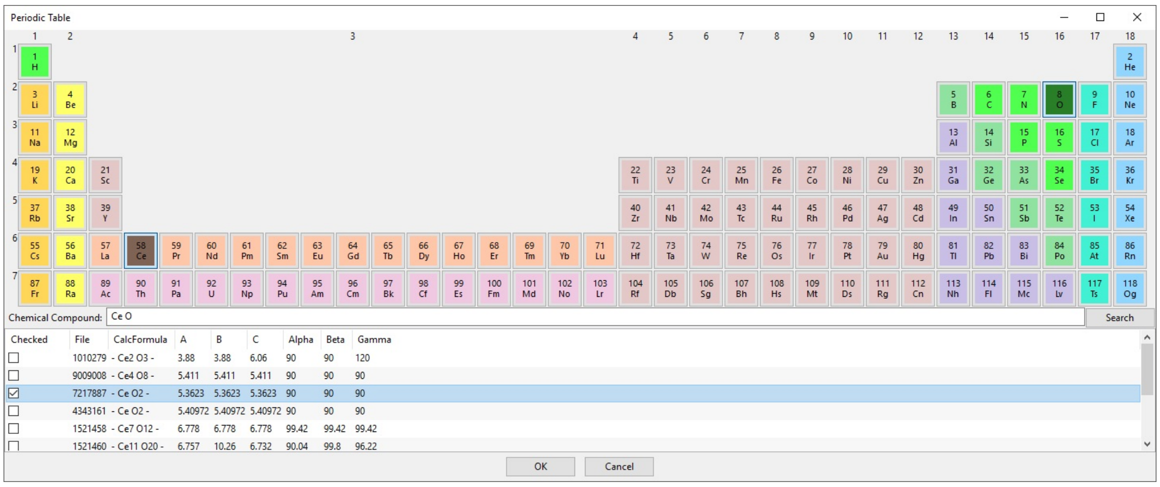
5. Preview 3D model of the selected CIF unit cell.

   This interactive model may be used to display some arbitrary combination of zone axis and upward vector by using the controls on the right side. Also, it may display the model with more than one unit cell in each dimensions, labelled as supercell.

   The main purpose for this model is to allow the user to confirm if the CIF loaded is the expected one. It also represents the model with a standard colour provided by Jmol software, which should ease the visual identification.

6. Selected CIF properties.

   This area is mainly used for CIF confirmation, as it displays the CIF file properties, unit cell information of its dimensions and angles, and displayed atoms.



Chemical Compound: Ce O

Checked	File	CalcFormula	A	B	C	Alpha	Beta	Gamma
☐	1010279	- Ce2 O3 -	3.88	3.88	6.06	90	90	120
☐	9009008	- Ce4 O8 -	5.411	5.411	5.411	90	90	90
☑	7217887	- Ce O2 -	5.3623	5.3623	5.3623	90	90	90
☐	4343161	- Ce O2 -	5.40972	5.40972	5.40972	90	90	90
☐	1521458	- Ce7 O12 -	6.778	6.778	6.778	99.42	99.42	99.42
☐	1521460	- Ce11 O20 -	6.757	10.26	6.732	90.04	99.8	96.22

Figure 9.: CIF search from COD database

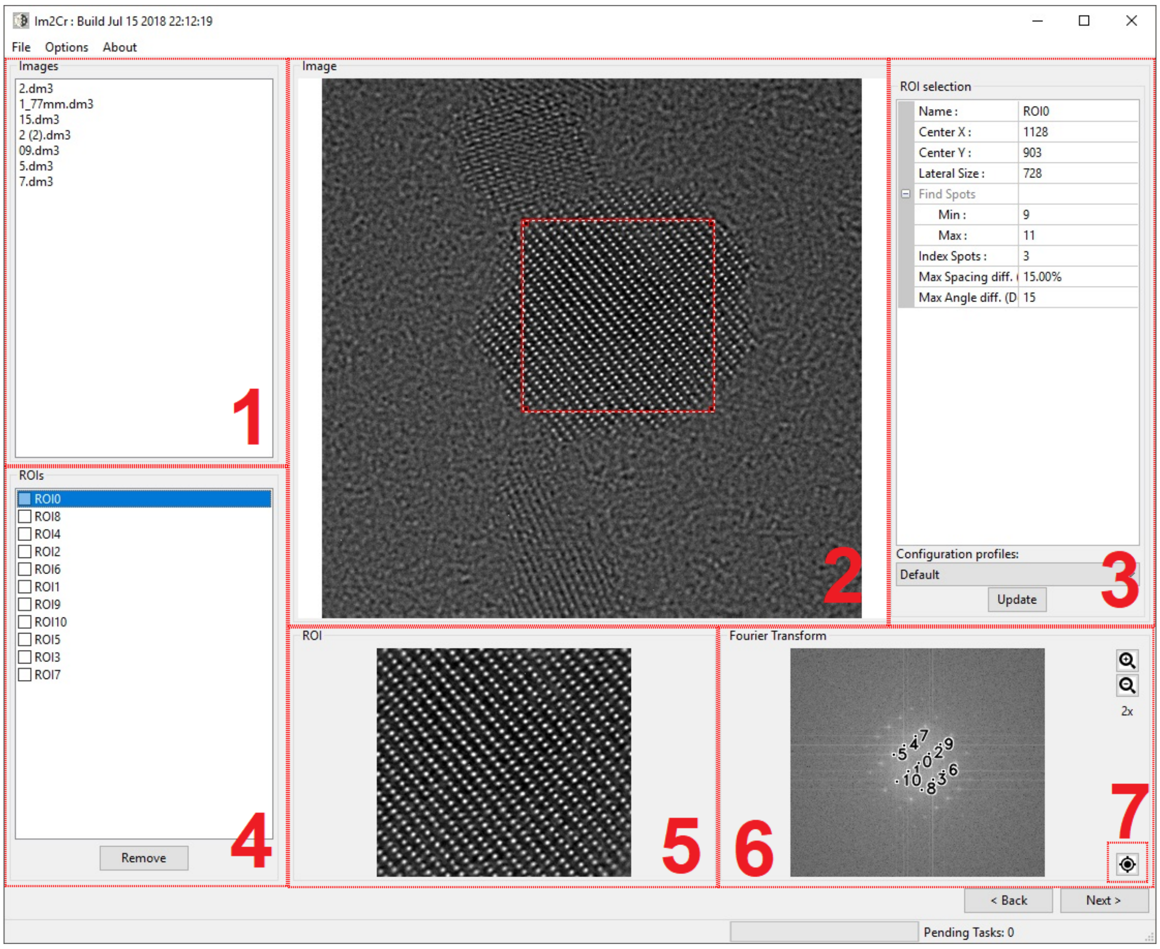
After setting all inputs, by selecting the next button, all the inputs are validated. In order to proceed to the next phase, there must be at least one image and one CIF. Also all the images must have their pixel size property. Failing to comply with these requirements will result in a warning box with the probable cause.

3.1.2 *ROI selection*

Once loaded all required files, the user is requested to select the desired ROIs in each image. The ROI will then be used to compute the Fourier Transform (FT).

An FT image is a visual representation of the information present on the original image by frequency. This image is used to identify frequency spots correspondent to the repetition of information on the image, i.e. unit cells of the supercell present on the ROI.

The spot identification and selection process is automatic. However, if the user is not pleased with the automatic selection, there is a functionality to modify the spots location.



2.dm3
1_77mm.dm3
15.dm3
2 (2).dm3
09.dm3
5.dm3
7.dm3

ROI0
ROI8
ROI4
ROI2
ROI6
ROI1
ROI9
ROI10
ROI5
ROI3
ROI7

Name :	ROI0
Center X :	1128
Center Y :	903
Lateral Size :	728
Find Spots	
Min :	9
Max :	11
Index Spots :	3
Max Spacing diff. (	15.00%
Max Angle diff. (D	15

Figure 10.: ROI selection

The interface displayed at image 10 shows an example of a ROI selection.

1. List of loaded images

   On the top left, is a list box with the loaded images on the previous screen. By selecting each image, the interface should be cleared for a new ROI selection.

2. ROI selection area

   In this area, the user should use its mouse to drag a box from one of the ROI's corner to the opposite one, being displayed a dashed red box previewing the ROI selected. Note that the box selection is only suited for TEM or STEM images. On the ED images, the selection is done by setting two opposite spots, being the centre the half distance between these two.

3. ROI selection details and indexing configuration

   On this property list the user may see the ROI selection properties. These include the ROI's name, centre and lateral size dimensions. There is also the possibility to modify these values directly, being the centre and lateral size dimensions reflected on the ROI selection upon modification.

   Bellow this information are indexing configuration properties as the number of spots to be selected and the maximum angle and maximum spacing allowed for the index process. These fields should be set with the preset values for the image type. This functionality is set in order to minimise user iteration with the tool, being these values the ones found to be best from the study of multiple images. These values are also configurable by a configuration file and may be changed by the user.

4. Selected ROI list

   After ROI selection and upon click on the Add button, the selected ROI's name is added on this list. Here should be all the selected ROIs to be indexed by the application.

   When selecting an item from this list, the interface will display the original image and the selected ROI, as well as the FT of the ROI and selected spots.

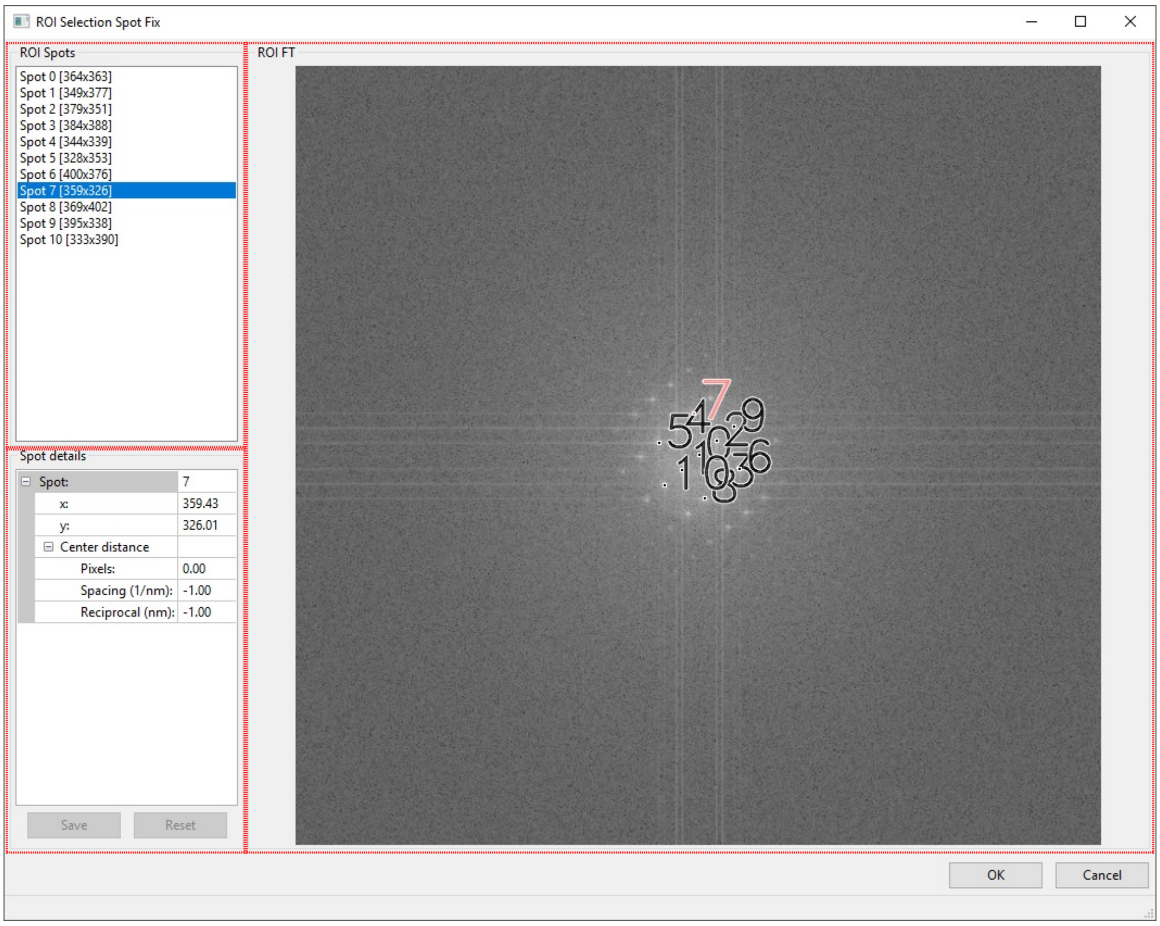
5. ROI preview

   On the bottom centre is a preview of the ROI selected. This is a representation of the image used to perform the FT. This image is changed when the user selects a different ROI or updates an existing one.

6. ROI Fourier Transform preview

   On the bottom right, similar to the ROI preview, upon ROI selection this area displays the cropped ROI FT. Also, it is possible to see which spots were detected on the FT. Note that the number of automatically detected spots are configured above on area 3, being the changes made reflected on the image displayed.

   For higher resolution ROIs, the FT spots may be too small on the preview. For this purpose, there is a zoom functionality, which enables up to four times zoom. Bellow these controls is the spot correction functionality seen on image 11.



Spot 0 [364x363]
Spot 1 [349x377]
Spot 2 [379x351]
Spot 3 [384x388]
Spot 4 [344x339]
Spot 5 [328x353]
Spot 6 [400x376]
Spot 7 [359x326]
Spot 8 [369x402]
Spot 9 [395x338]
Spot 10 [333x390]

Spot:	7
x:	359.43
y:	326.01
Center distance	
Pixels:	0.00
Spacing (1/nm):	-1.00
Reciprocal (nm):	-1.00

Figure 11.: Spot correction

7. ROI selection spot fix

If the user does not agree with the automatic spot selection or wants to make adjustments on the spots, the interface on image 11 is available by clicking on the target icon. In this interface, the user may select from the list of spots detected the one to be modified and, either by using the mouse and clicking on the image or by introducing the pixel coordinates on left, adjust the spot location.

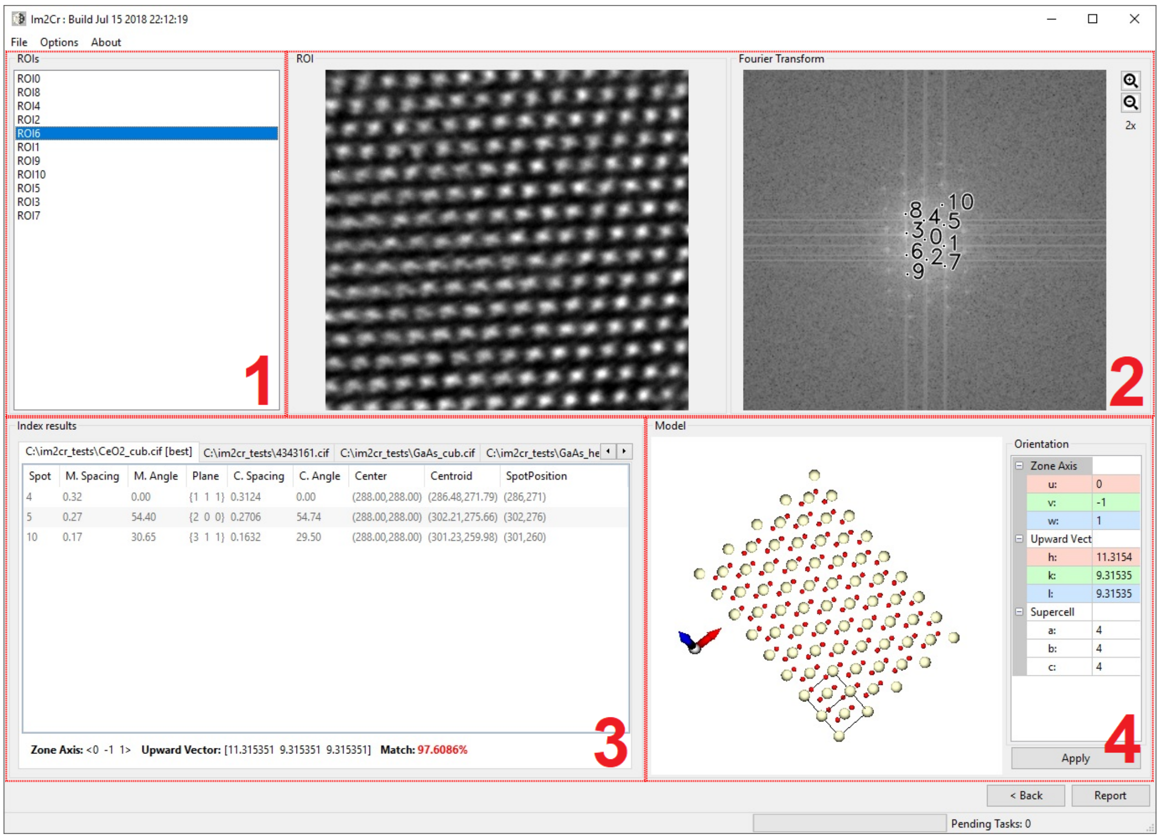
Note that the spot is displayed with red accent in order to be easily identifiable.

Once all modifications are performed, the user may select save or revert the changes made by clicking on reset. All the changes made are reflected on the ROI selection interface.

After adding the ROI, the user may notice the ROI's name being update on the list. First it appears the suffix "Loading" and after some time, without it. This is an optimisation introduced that, when the ROI information is complete, the indexation process begins on background while the user is free to select more ROIs.

Once all ROIs are selected, the user may proceed to the last interface, which displays the best indexation found for each ROI.

3.1.3 *Index results*



C:\im2cr_tests\CeO2_cub.cif [best] | C:\im2cr_tests\4343161.cif | C:\im2cr_tests\GaAs_cub.cif | C:\im2cr_tests\GaAs_he

Spot	M. Spacing	M. Angle	Plane	C. Spacing	C. Angle	Center	Centroid	SpotPosition
4	0.32	0.00	{1 1 1}	0.3124	0.00	(288.00,288.00)	(286.48,271.79)	(286,271)
5	0.27	54.40	{2 0 0}	0.2706	54.74	(288.00,288.00)	(302.21,275.66)	(302,276)
10	0.17	30.65	{3 1 1}	0.1632	29.50	(288.00,288.00)	(301.23,259.98)	(301,260)

Zone Axis: <0 -1 1> **Upward Vector:** [11.315351 9.315351 9.315351] **Match:** 97.6086%

Orientation

Zone Axis	
u:	0
v:	-1
w:	1
Upward Vect	
h:	11.3154
k:	9.31535
l:	9.31535
Supercell	
a:	4
b:	4
c:	4

Apply

Figure 12.: Indexation results

The last interface page is dedicated to display the index results. As displayed on the image 12, the user may see the best matching CIF's unit cell with its zone axis and upward vectors highlighted on the bottom left.

The interface layout may be splitted into four main areas.

1. List of selected ROIs.

   This list is the main control for displaying index results. By selecting different ROIs, the other areas are refreshed accordingly. This list box is the same present on the previous interface after adding a new ROI.

2. ROI and FT spots display.

   These images are also present in the previous interface. The main reason to also display them in this interface is so the user can compare the simulated 3D model to the real image. Also, the FT image with spot numbers have a special importance, since the numbers displayed are directly correlated to the spot number on the index table.

3. Index result table.

   This area is where the main focus of this software is. At this table, the indexed CIFs are ordered from left to right by their matching rate, being the first tab the best indexation found. Regarding the table, this contains FT spots with their miller indexes, experimental angle and measured d-spacing values from the image and the theoretical values computed in the indexation process for each CIF.

4. 3D model

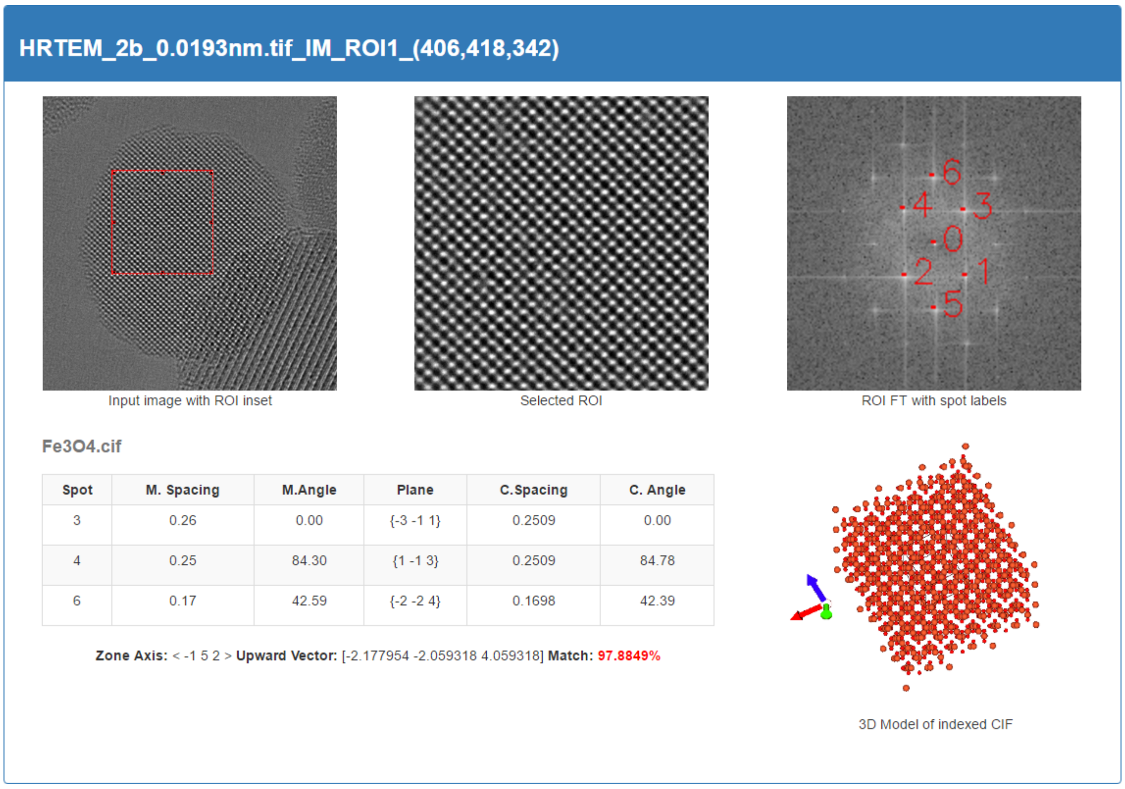
   At the bottom centre is the 3D model representation of the CIF's supercell in the orientation given by the index process. Note that these values are also configurable on the right by the model properties where the user may change the supercell dimensions, as well as try other orientations.

   The objective of this 3D model is to compare the theoretical best CIF found with the ROI image above. If the experimental image is representative of the atomic structure, then this model should display the same pattern.

Once the user agrees with the indexation performed, he may export the images and a print of the 3D model by clicking on the Report button. This will produce an HTML page with the indexation table and the images. This will also export an XML file with the information present in the indexation table, so that other applications may use this output for other purposes.

3.1.4 *Report*

At the end of an indexation session, the results are exported into a printable HTML report page, similar to the image 13. This page contains all the important information displayed on the last application interface.



HRTEM_2b_0.0193nm.tif_IM_ROI1_(406,418,342)

Fe3O4.cif

Spot	M. Spacing	M.Angle	Plane	C.Spacing	C. Angle
3	0.26	0.00	{-3 -1 1}	0.2509	0.00
4	0.25	84.30	{1 -1 3}	0.2509	84.78
6	0.17	42.59	{-2 -2 4}	0.1698	42.39

Zone Axis: < -1 5 2 > **Upward Vector:** [-2.177954 -2.059318 4.059318] **Match:** 97.8849%

Figure 13.: Report exported

One requested functionality was to not limit the use of the images to the report page only. As such, when exporting a report, the images used are also exported with their full resolution, allowing the user to use them for other purposes.

From the end user perspective, this completes the software overview. The next section will cover in more detail the application design from the software development view.

3.2 APPLICATION DESIGN

The development of a software tool involves a careful study of many factors, from the user interface to the code language used. One of the most important phases of the development is the software design, where all the application components are specified and the interaction between them drawn.

This section will present the application structure by first looking at the components present and latter to the interactions between them.

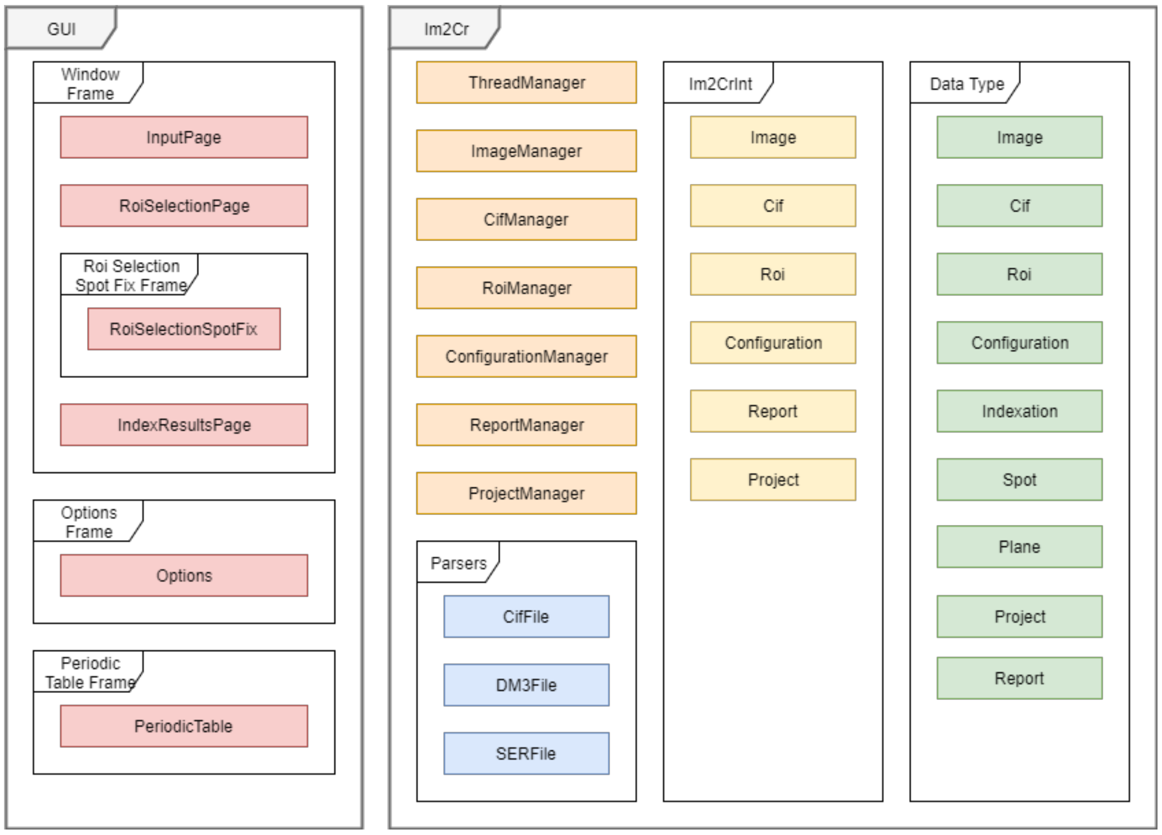


Figure 14.: Overview of the application structure

3.2.1 *Application overview*

When designing the internal structure of the application, a major concern was to be able to reuse the application logic with multiple interfaces. In order to accomplish this, the Im2Cr implements the Model–View–Controller (MVC) architectural pattern to separate the interface from both data and application logic. In the image 14 it is possible to see this separation by the GUI and Im2Cr packages.

Regarding the user interface, this contains all the main interfaces displayed to the user. Each one of them contains the interface components as buttons, list boxes, image, etc, and the controllers for each input by the user.

For the Im2Cr package, the application core, the goal here was to be able to accept inputs from multiple user interfaces. As such, the data and execution logic are all contained within this package. To input/retrieve data and perform operations, ”Managers”, which are coloured in orange, were created. These are an implementation of the Facade pattern. The objective is to have controlled access points to the application, which hides all the inner logic. This encapsulation in turn allowed to introduce code parallelisation and optimisations without changing the interface code, making this design modular and giving more freedom for development.

Data exchange between the interface and the core is performed by the included data types, coloured in green. Each data type has been modulated into an object with the respective information stored on it.

To be able to use the same core with other interfaces, this was encapsulated into a library project, which sets a clear separation between interface and core logic, at the cost of better application optimisation.

3.2.2 *Manager execution*

Figure 15 gives an overview of the Manager execution interaction.

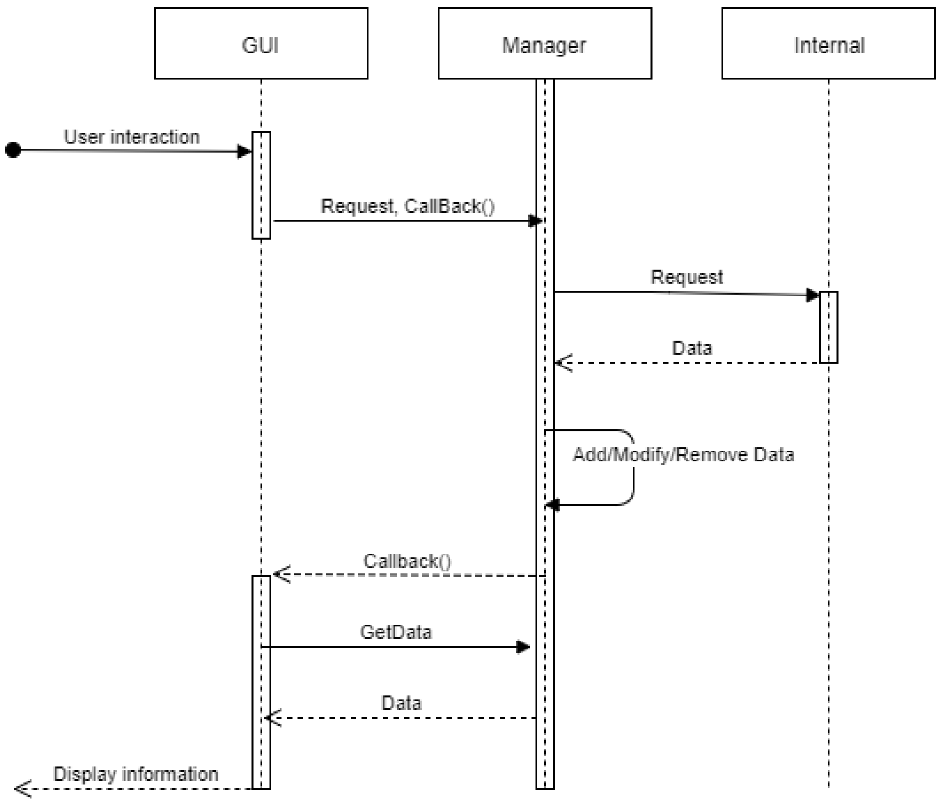


Figure 15.: Generalisation of a manager execution

The application to be developed should try to meet all the end user requirements while maintaining a good performance. Nowadays, the computer performance technology has shifted their paradigms from increased CPU clock frequency to increased number of available processor units, or native threads. While this could mean that the overall application performance has the potential to be increased by each added processor unit, the same cannot be said for the real-world performance of existing software. In some cases, the performance of older software may even be equal or worst on modern CPU.

The fundamental problem is that most legacy software was programmed to execute in serial, using only a single processor which have a strong dependency from its clock speed. In the present time, the way to extract the maximum performance available is to use all the processor units at the same time. To achieve this, it is required to write parallel code and minimise code dependencies.

Taking this into consideration, the design of Im2Cr started from the perspective of maximising the execution in parallel.

Starting with the graphical user interface, the user should not perceive the interface as ″frozen″, which means that the thread used to run the interface must be detached from all the execution at all time and kept as lightweight as possible.

Having the interface not waiting for the input results, this means that inner logic execution must be performed in parallel to the interface execution.

To keep the code organised and take advantage of available resources on the computer, every time data is requested, their respective Manager is called. Figure 15 is an example of how this process occurs.

When the user triggers some action which requires data, the interface executes a set of its own validation and procedures and calls the respective Manager. As example of this action is when the user adds a new image file. In this case, the interface gets a file path, which in turn asks the Image Manager to read the file at that location. While requesting the file load action, it also sets the procedure to execute once the file has been loaded, typically an interface refresh. This procedure is also known as a Callback function, which can have its own arguments. After making the request, the graphical interface is free to receive other user inputs, while the execution proceeds on background.

After receiving the request, the manager calls a set of internal functions. The reason for this higher-level encapsulation is to make the API more readable by keeping the code simple and maintainable. Once the internal component retrieves the data, this is added/-modified/removed from the manager data structures and the callback function is called. For instance, in the case of an input image, while it is being loaded the image name in the interface has the suffix ″loading″. Once the image is loaded and the callback function called, this function removes the ″loading″ word, which tells the user that the image is ready to be used. Also in this example, if the image loaded is to be displayed, the image automatically appears on the interface.

This ″Manager″ based design requires all the main objects in the application to have its own manager. Also, these managers may be used on different interfaces, which could create a problem of different instantiations for each manager and, in turn, with data spread by each instance.

To solve this problem, all Managers have been based on the Singleton pattern. This pattern makes a class to only have a single object instance across the program. With this implementation, all the data structures are shared between all objects with references to the manager class. This creates the possibility of this application to be executed on a server where there could be many client interfaces, all sharing the same data.

3.2.3 *Parallel execution*

Until now parallel execution was referred as background execution. This was because there are many ways to have parallel execution, from within the processor to across multiple computers connected on a network. However, not all of them are well suited for any kind of application. For this reason, there must be a study on which kind of parallel execution the application should implement.

For this application, the work performed revealed to not be too computational sensitive in the sense that the execution of most procedures took a minimal amount of time. For this reason, the distributed parallelisation on a network have been excluded, since it would require a significant amount of time to send information across the network.

Even within the same computer, there are different approaches to be taken based on the work to be performed. For example, simple tasks applied to multiple data could benefit from the parallelisation using accelerators as the Graphics Processing Units (GPU) or Intel Xeon Phi. The last level and most simple parallelisation design is to use all cores available in the CPU. This last one revealed to be the best suited for this application use case, given the hardware it is expected to be executed, a desktop/laptop.

Regarding the parallelisation within the CPU, the most basic one is by using the available threads, assigning each task to a single thread. However, if many tasks are expected to be executed, the time taken by thread creation, deletion and context switch could have a significant impact on performance.

For this reason, the parallelisation implementation in this application is based on a thread pool, which combines the execution on multiple threads, while keeping the cost of creation and deletion to a minimum. Also, the use of a thread-based parallelism still has costs associated with context switching, which occurs every time a thread is set on hold in favour of another to be executed. When this occurs, the variables used by the replacing thread must be loaded into memory. To remove even this overhead, the approach taken was to use task parallelism Intel (2012), in this case, the Intel Threading Building Blocks (TBB) library implementation. This kind of parallelism assigns a task to a thread (which could already have other tasks assigned) and have the threads allocated on a thread pool.

The parallelism implementation in the application occurs mainly on file inputs, both images and CIF files, and on the indexation of all loaded CIFs. All these tasks are fully independent, having the only dependency the addition/removal from the data structures on the managers. These operations are performed atomically, which should keep the data structures consistent and solve concurrent access.

3.3 IMPLEMENTATION

With the user requirements and the application layout set, the next phase in the application development is the individual component implementation.

In this section is described in detail the algorithms and logic applied to each component in the application.

3.3.1 *File inputs*

The first step taken by the user with the Im2Cr is to load images and CIF files. Since the target audience for this application is somewhat limited, the number of open-source and license free libraries are scarce or non-existent. For this reason, some of the file formats had to be implemented in order to extract the required information.

In the context of the Im2Cr application, the image formats supported are SER, DM3 and TIFF. This last one is currently supported by the image processing library used, OpenCV, which simplifies the implementation. However, the same cannot be said for the other two image formats. For these, a custom implementation had to be made. For the Crystallographic structures, a custom CIF parser was also implemented.

Following is a brief description of the implemented parsers and some detail about their internal structure.

3.3.1.1 *SER format*

The SER file format is a proprietary format and was developed by the Emispec company. This company has now been bought by FEI, another microscope company, but this file format is still one of the most used file formats.

Since this is a proprietary file format, the implementation of a parser for this file had its specifications taken from community shared information[1].

This format has some flexibility in allowing up to 4 dimensions of data to be stored. Its structure is comprised by six parts, being the first the header, followed by dimension array, data offset array, tag offset array, data elements and data tags.

For Im2Cr, the useful information is: (i) the ”CalibrationDelta”, with the pixel size in meters; (ii) ”ArrayLength”, which contains the lateral size of the square image; (iii) the

1 TIA (Emispec) file format: http://www.er-c.org/cbb/info/TIAformat/

"DataType", which sets the image pixel encoding in bytes; and (iv) the variable size "Data", that stores the image information.

3.3.1.2 *DM3 format*

The DM3 file format is also a proprietary format used by the Gatan company. This format is greatly more complex when compared with the above SER format, but much more flexible, as the information is layout in a tree format, having each node a name associated with the information contained.

The format in itself is quite simple, having only an header, a tag directory which contains *n* tags. Each tag could be a simple value, an array of values, a group (struct) or an array of groups.

Because of the flexible tree structure, it is not possible to point where the image information is without reading all the file. According to the community file specification page "There is no simple way of finding the length of a type 15 tag without completely decoding it and working out the number of bytes in each data type"[2]. This means that DM3 images imported on the application must have their entire file parsed before being able to get the image information.

In the Im2Cr context this implies that, once the file is loaded and parsed, the tags must be searched in order to find where the image data and other useful information is. In the Im2Cr, the important tags are: "LowLimit" and "HighLimit" with the values for the lowest and highest pixel intensities, the "Width" and "Height" of the image inside the "Dimensions" tag, the "Scale" inside "Calibrations" with the pixel size information, and both the data type and image information inside the "Data" tag.

Once the image information is read into an openCV Mat structure with the informed data type, the image is normalized with the low and high limit values.

3.3.1.3 *CIF format*

For the crystallographic information there was a need for a CIF format parser. This format is used globally as a way to describe crystallographic information and other related information, as the author and publication.

This format has a very simple text-based structure, containing usually a single block which has either single data values or multiple data values. The first one can be thought of as a simple ¡key, value¿ pair, while the second is like a table, where the headers are specified on the start and then each row contains data.

The best and worst of this file format is its flexibility. There is no fixed structure specified, nor the tables have the number of elements set. This makes the parsing of a CIF file somewhat complex, by always having to search the following line and then decide if it is

2 Digital Micrograph file format: http://www.er-c.org/cbb/info/dmformat/index.html

still an element of the table being parsed, a comment, another table begin or a single data value. Even the single data values can have their value on the same line or on the next line.

In the Im2Cr, all CIF files are loaded into memory and only then the properties are read. The extracted information includes: *a*, *b* and *c* distances related to the dimensions of the unit cell, *alpha*, *beta* and *gamma* angles between axis, the table of atoms and their initial location, and a table of atom's simmetries.

Unit cell representations of CIF files make use of the *a*, *b* and *c* dimensions multiplied by fractional coordinates[3] (normalized location between 0 and 1) of each atom and applied to each symmetry on the table.

3.3.2 *Selection of ROI*

The ROI selection process still is one of the most challenging tasks from which to remove the human intervention. In this process, the user is required to select the ROIs in the image.

From the computational standpoint, some preliminary work was made in order to automate this process. The work done relied on the higher pixel contrast each ROI had in comparison with the remaining amorphous regions on the image. Taking this into account, the designed algorithm scanned the experimental image by its rows and by its columns and built a graph with the variance of each dimension. Once the graph was plotted, then the algorithm tried to fit a polynomial function of a fifth order. The reason for the fifth order polynomial was to have the best possible fit and to have represented four inflection points. With this information, then the variance from the maximum and the minimum values of this polynomial was measured and, if it was considered relevant (by a configurable parameter), a water line was set. Using this algorithm, only the middle curve above the water line was considered as a ROI. This implementation may be seen in images 6 and 7.

The study of a test set of images showed promising results, having this algorithm selected good approximations to the real ROIs. However, this algorithm had a significant flaw in its conception that was to assume that only a single ROI was displayed at each image.

Since the state of the art on the electron microscopy reveals that microscopes are capturing increasingly higher resolution images, the prospects for image indexation is to have multiple ROIs displayed in a single image. For this reason, the preliminary works described above had to be halted for future improvements, since its current state was still considered unreliable.

To overcome this limitation the human intervention is still required to select the ROI on the image. With the direct user input being required, the software design had to take this into consideration, making the user experience one of the main focus areas.

3 A Guide to CIF for Authors: `https://www.iucr.org/__data/assets/pdf_file/0019/22618/cifguide.pdf`

The solution found was to make a GUI and allowing the user to draw with the mouse the ROI area on the image. This solution however came with other related problems, the arbitrary actions made by the user. In the particular case of ROI selection, it was necessary to limit the area drawn to be squared. This limitation was necessary mainly because the distortions present on a non-squared Fourier Transform, described on the following section.

3.3.3 *Fourier Transform*

After the image was cropped into a ROI, the next step is to study the atomic periodicity in the image. The objective is to compute the distance between atoms and their relative angles. For this task, a widely used operation is to obtain a Fourier Transform of the image.

The FT operation converts the spatial information into the frequency domain in the form of waves. Since a wave is represented by its phase and amplitude, a common representation for this information is by using complex numbers.

In the case of images, the input spatial information is in two dimensions. As a result, the output transformation is a two-dimensional matrix of complex numbers. In order to visualize the FT result, there is a need to transform two-dimensional matrix of complex numbers into a two dimensional matrix of real numbers, to be later displayed as an image. This transformation is made using a modulus computation of each complex number.

From the experience taken from developing the software tool Peakfinder Silva and Sobral (2015), the majority of the atomic information is given at the higher intensity pixels on the FT image, namely the ones closer to the centre but not including the centre. On the Peakfinder tool, the goal was to remove the amorphous background from the image, highlighting the atomic structure. For this task a pass-band filter was applied, having as high pass filter the radial distance after which the radial variance of pixel intensities no longer had significant peaks or had already been selected the desired number of peaks. As a low pass filter, an approximate position between the centre and the first radial of FT spots. With this experience, it was clear the FT peaks did contain useful information about the structure periodicity.

On the Im2Cr tool, the approach taken was a little different, as the goal was not to remove the background noise on the images, but to locate the FT spots related to the atom's positions. For this task, the objective was to perform image segmentation by selecting only the FT spots on the FT image.

Image segmentation is widely used as a step to remove background information from foreground information. The core of this algorithm can be summarised in considering as background the pixels which are bellow some reference value and foreground the pixels equal or above this value. Note that there is a need to input the threshold value which was considered as the reference value.

On the developed tool, the manual input from the user about the threshold level was impractical, as this would likely translate into a trial and error process until the user had the desired spots located. Instead of this approach, the algorithm introduced use the desired number of spots to be located as the user input, making the determination of a threshold value an automated process.

3.3.4 *Autonomous spot detector*

Once the FT was applied to the experimental image, the objective was to locate the FT spots displayed. For this task image segmentation should be performed in order to extract the FT spots from the image. Since the user is not required to input the threshold value to be used, another kind of input is needed. For the Im2Cr, the input required from the user is the interval of spots to be detected on the image.

The algorithm used consists of performing image segmentation with an initial configurable value and count the number of spots detected. If the number of spots is within the spot interval given by the user the algorithm stops, otherwise the threshold value is increased or decreased depending if the number of spots found was respectively bellow or above the interval. This process is repeated until the previous condition of the number of spots is within the interval or the number of iterations is greater than a configurable number of maximum iterations, to avoid the application to be kept in a possible loop.

Describing the segmentation process in more detail, this process starts by applying the threshold value to every pixel on the image, marking every pixel with a flag of background or foreground if the pixel value is below or above the threshold value. Once this process is completed, the next step is identifying spots of foreground pixels. For this task, the algorithm starts by mapping all pixel coordinates with a flag "not visited" and searching from the beginning for foreground pixels, marking all pixels along the way as "visited" if they are background pixels. Once a foreground pixel is found, the flag matrix switches the pixel position to "queued" and searches is neighbour pixels for more "not visited" foreground pixels. To each neighbour foreground pixel, the algorithm marks its position as "queued" and moves to its location to search for more foreground pixel positions. When no more foreground pixels are found, the algorithm starts popping "queued" pixel positions and adding them to a spot structure. While this process is done, all popped pixels are marked as "visited", having this flag the objective of its position to not be searched again or included on another spot structure. With this algorithm, all pixels are visited only once.

When the spot location algorithm finishes, the result is a list of spots structures, having each one of them the original pixels location. With this information it is possible to compute the centroid of each spot. For this computation, the polygon centroid formula was used, which requires the coordinates of the polygon perimeter. In this case, the polygon is the spot and its perimeter the outer pixel positions.

Once this process is applied, the result is the number of spots detected and its centroid positions.

However, if the first threshold value did not produce the expected number of detected spots, a correction should be made to this value. For this task a new algorithm was introduced to autonomously find the best threshold value.

The developed algorithm starts by requiring the information about the previously number of spots detected in relation to the expected interval. If the number of detected spots was above the user expectations, a more restricting threshold value should be used, otherwise a lower threshold. Regarding the new threshold value, this is computed from a variation of the binary search or half-interval search algorithm. On the original algorithm, for a given initial value on an interval and a target value on a sorted list, the algorithm splits the interval in halves in the direction of the targeted value until the value is found.

On the Im2Cr application, the implementation of this algorithm for threshold determination was not suitable. The non-suitability of this algorithm was found after studying the behaviour for a set of test images. In this algorithm, if the first threshold value had not found the number of expected spots, most probably the second iteration would not find it as well, because the next threshold value would be too high or low with relation to the final threshold value.

To overcome this inefficiency, some adaptation was needed, in this case, the interval reduction. On the original algorithm the search interval is always reduced to half the original size and the new value is at the middle of the new interval. On the implemented adapted version, the interval reduction is configurable, being found that a 90% reduction at each iteration made the new value much closer to the final threshold value, reducing as well the number of required iterations and consequently, the time taken by the application to execute.

3.3.5 *Angle and d-Spacing computation*

When the application finishes executing the automatic threshold algorithm, the number of spots, its location and its centroid are known. However, the centroid position was based on a computed value, which may not correspond to the real centroid due to the margin of error introduced by the image segmentation process. Taking this into consideration, there is a need to compare the computed centroid to the highest intensity pixel on the delimited spot, which indicates the real centroid. Most of the times, the computed centroid corresponds to the real centroid position however, if the spots are scattered over a wide area or if in one dimension the frequency spread is wider than the other, the computed centroid may be slightly misaligned. In these cases, a centroid correction is performed in order to switch its position to the correct one.

Having performed a centroid correction, enables the computation of d-spacings and the relative angles from the spot to the centre position. To compute the d-spacing of each spot, first should be done a scale calibration. This calibration may be needed if the image was cropped, as the reduction of the original lateral size on the original image corresponds to its inverse on the FT image. So the pixel size on the cropped FT should be recalculated from the lateral dimension of the cropped ROI, given by the multiplication of the pixel size by the number of pixels on one dimension. Note the ROI image is a square image, so the pixel count is the same for both width and height.

Once the new pixel size is computed, the d-spacing is obtained by multiplying it by the distance in pixel from the centre to the spot centroid. This distance is a simple computation of the hypotenuse of a triangle made by the centroid coordinates on the image in relation to the FT centre coordinates, considered as the origin.

For the computation of the angle, first it should be set a reference point to which the angle is zero. In this application it was considered as a reference point the closest spot from the FT centre (known as reference point). Then, each spot centroid angle is computed from the angle formed by the reference point, the FT centre and the centroid position.

From this point forward, the d-spacing and angle values computed above will be considered as experimental values, as these have direct correlation to the input image. These values will later be compared to the theoretical ones in the indexation process.

3.3.6 *CIF indexation*

Once the angle and d-spacing of each spot have been computed, the next step is the comparison of these values with the expected values from different unit cells.

The information about each unit cell is present in a CIF file. The CIF file contains a standard structure mainly used to describe the crystallographic information about each unit cell, given by its atomic composition, the relative position of each atom, the unit cell lateral dimensions and angles in a three-dimensional space, and many other optional information, from the original authors to the journal it was published on.

For this application, the relevant information to begin the indexation process is the unit cell lateral dimensions (a,b and c) and its angles (α, β and γ).

This process starts by computing the theoretical values of d-spacing expected to a given $\{h,k,l\}$ plane. Since at this point it is unknown the best matching plane for each spot, the theoretical value should be computed to a set of expected planes. In the Im2Cr application, it is computed all combinations with values from -4 to 4 in each component, being this value configurable. The formulas used to obtain the d-spacing value are directly related to the unit cell geometrical form. So, a cubic structure has a different formula than a tetragonal one for example. To overcome this limitation, the Im2Cr implements the most generic formula that could be applied to any geometrical structure, known as triclinic. Even if this formula requires more computations, the computations needed for geometrical determination of the input unit cell structure could made the written code more prone to errors and more complex. Giving as input the unit cell dimensions, its angles and a plane, the output is the theoretical d-spacing for that plane. This process is repeated for every expected plane.

At this point, the developed application has a set of planes and its d-spacings to which the experimental ones can be compared. To give a sense of the number of computations required in this process, the option to study all planes from -4 to 4 for each *h*, *k* and *l* component result in 728 possible combinations, excluding the origin point. If the application were to compare all these possibilities to all the spots detected, the number of combinations grow exponentially with the number of spots to be indexed, making the indexation process unfeasible and unscalable for multiple spots. To address this problem, the only {h,k,l} planes considered are the ones that have a theoretical d-spacing between the maximum and minimum experimental d-spacings. To this interval of values it is added a configurable relative margin to the top and lower endpoints values. After the possible planes are selected, each spot is given its own set of possible indexable planes.

Indexing only by the best matching d-spacing could result in erroneous results. For instance, in a cubic form, there are multiple planes with the same distance from the centre. To overcome this problem, the angles should also be part of the indexation process.

To compute the angle between planes, the geometrical shape should be taken into account. Like the d-spacing computation, for angles between two different planes there is a triclinic formula, which outputs the cosine of the angle. Since the interest is on finding the angle, to this value it should be applied the arc cosine.

With the angle computation, the indexation process may now proceed to find the best matching plane to each spot. The selection formula used (1) is custom made to have more sensibility to variations on the angles than the d-spacings.

$$Classification = \prod_{i=0}^{N}(1 - SpacingDiff_i) * \left(\frac{180 - AngleToRefDiff_{0,i}}{180}\right)^2 \tag{1}$$

$$SpacingDiff(i) = |MeasuredSpacing_i - ComputedSpacing_i| \tag{2}$$

$$Spacing = \frac{1}{\|\vec{p}\| * CalibratedFT} \tag{3}$$

$$\|\vec{p}\| = \sqrt{(centre_x - point_x)^2 + (centre_y - point_y)^2} \tag{4}$$

Being:
p - Length vector of the vector formed after subtracting the centre to the point location

$$CalibratedFT = \frac{1}{PS * LR} \tag{5}$$

Being:
PS - Image pixel size in nm
LR - Number of pixels in one dimension

$$AngleToRefDiff(\vec{r}, \vec{v}) = \arccos \frac{(\vec{r} \cdot \vec{v})}{\|\vec{r}\| \cdot \|\vec{v}\|} \tag{6}$$

Being:
r - reference vector
v - vector
The decision to differentiate d-spacings and angles was taken because d-spacings are more prone to variations due to its low precision than the angles, which should not vary as much. Using this formula, the values should be between 0 and 1, being 1 the best indexation possible.

To be noted however that the classification formula is applied for a single plane for each spot. While indexing each spot has its list of plane candidates. This makes the classification computation to be performed multiple times for each combination of spot plane candidates before the best indexation is found.

3.3.7 *Zone axis and upward vector determination*

While performing plane indexation to each spot, the angle formed by the planes indexed should also be take into consideration, as this has a major impact on the zone axis determination.

The zone axis computation basically uses the {h,k,l} components of each two plane candidates and compute the matrix determinant in order to find the third {h,k,l}, or in this case the zone axis.

For index consistency, all two spot combinations from the number of spots selected to be indexed should have the same zone axis. Having this condition makes the plane combinations that do not comply to be rejected as solutions.

$$UV = n * A + m * B \tag{7}$$

Once the zone axis was successfully found, the next step is to find its upward vector. This computation starts by using two spots and the angles formed between their location and a reference point. The reference point is an arbitrary location on the image that should be applied to all angle computations. In this application, it was chosen the point at the height centre and full width, being this aligned in the height dimension of the image centre. With the spots and their angles, it is applied the linear combination of the spots locations represented in the formula (7) to find the proportion of each component that gives the vector pointing up on the image.

$$n = -1 * \left(\frac{Spacing_{spot2}^{-1} * \cos\beta}{Spacing_{spot1}^{-1} * \cos\alpha} \right) \tag{8}$$

Since the interest is to find the proportion, we can set one of the constants to 1 so that we can determine the n constant. The formula (8) represents the computation needed to be performed, being α the angle formed from the reference point and the first spot, and β the angle formed between reference point and the second spot, both at the image centre.

After the n value is found, it can then be applied to the planes that represent each spot, giving the upward vector.

3.3.8 *Model render*

The last step is to visualise the best matching CIF file with the indexation information taken from the previous step.

For this the application displays the best CIF indexation found for each ROI using the CIF symmetries provided. Along with atomic representation, it is also needed the camera position and the upward vector of the model. Both of this are already known from the previous steps, so that the 3D representation uses these for the model orientation.

Related to the atomic representation, the CIF file content does not have any information regarding the atom size nor its colour. To overcome this problem, the application makes use of the database information taken from the JMOL software. This application is already well known in the crystallographic research area, so the model representation should have colours familiar to most users.

3.4 TOOL PORTABILITY AND LICENSING

Regarding the Im2Cr implementation and portability, this application aimed to be used by many researchers. As a result, the environment in which the tool may be used could be the most diverse possible. Also taken into consideration was the possibility of the developed application being integrated into other applications. With this in mind, a special care was taken when selecting libraries to be used on the Im2Cr.

In relation to the Operating System (OS) supported, the objective was to support all major OS systems: Windows, Linux and Mac OS. To reduce the need for OS specialization, the GUI library should be portable to these OS. The two different libraries which offer this functionality was the Qt and the wxWidgets. Being the first limited commercially, the choice was for the latter.

For the image edition, the mainly used library on C++ is the OpenCV[4] library. This library has ample documentation and a wide community supporting the project. It is also known for being well optimised for image processing.

For functionalities like thread management, thread safety and some other file system operations, it was used the Boost[5] library. This library is also well known for its care about application performance while being open source.

As for the configuration files management, these are XML based files. As a result, a XML library was selected to parse these files, being chosen the PugiXML[6] for its light-weight and great performance while sacrificing the validation functionality. Since the files used are configuration files automatically built by the application, this limitation was not considered significant.

The symmetries computation from the CIF file was relegated to the MuParser[7] library. This library takes variable values and a mathematical expression and outputs its value. This was useful, since it was not necessary to spend time parsing the symmetries present on the CIF file.

Lastly, the ROI selection through a selectable GUI rectangle was made using the wxRect-Tracker[8], more specifically the version included on the RPhoto open source application.

4 OpenCV https://opencv.org/
5 Boost C++ https://www.boost.org/
6 PugiXML https://pugixml.org/
7 MuParser http://beltoforion.de/article.php?a=muparser
8 wxRectTracker http://www.lprp.fr/soft/rphoto/wxrecttracker/index_en.php3

4 VALIDATION AND PERFORMANCE ANALYSIS

After the development phase the application should be tested and the performance tuned to the targeted systems. The application functionality is validated for several computing platforms, mainly those where it is expected to be executed.

This chapter starts to describe the test environment and used methodology to run the tests. The tests results are recorded and later further analysed. The application performance is then evaluated, where the parallel optimised version is compared with the original application performance.

4.1 TESTBED AND METHODOLOGY

The validation and evaluation of the application tool performance, in several computing platforms, require a fixed data set and a normalised methodology.

The validation methodology was based on a static test dataset to give equivalent results on all computing systems. The selected test dataset contains 7 images and 12 CIFs. Some of the images contain more than one ROI, leading to a total of 11 ROIs to test.

The metric of interest in this validation was the average time taken to process each dataset. This should represent the expected times from the application in a real-world scenario. The median is also displayed as a metric representative of the application execution time. This metric should avoid outliers and better represent the application performance.

Since the main target platform for this application are researchers laptops or laboratory computers with consumer oriented parts, the selected testbeds used available systems: two legacy systems, a desktop and a laptop with 2012 CPU models, and a more recent enterprise level equipment, a laptop with a 2015 CPU model. This choice of equipment aimed to give the expected results from the perspective of broadly available ageing systems.

The key specification data for the selected equipment follows:

Desktop 1:

- CPU: Intel Core i5 3570k @ 4.2GHz
- Cores/Threads: 4/4
- Cache: 6MiB L3
- Vectorization: SSE4.2, AVX
- Memory: 16GiB DDR3 @ 1600MHz (dual channel)
- OS: Windows 10 x64 (10.1803)

Laptop 1:

- CPU: Intel Core i7 3610QM @ 2.3-3.3GHz
- Cores/Threads: 4/8
- Cache: 6MiB L3
- Vectorization: SSE4.2, AVX
- Memory: 6GiB DDR3 @ 1300MHz (dual channel)
- OS: Windows 10 x64 (10.1803)

Laptop 2:

- CPU: Intel Core i7 6820HQ @ 2.7-3.6GHz
- Cores/Threads: 4/8
- Cache: 8MiB L3
- Vectorization: SSE4.2, AVX2
- Memory: 16GiB DDR4 @ 2133MHz (dual channel)
- OS: Windows 10 x64 (10.1709)

The measured values were taken on three runs, being the final result the average execution time of the three runs. This number of runs will try to minimize potential outliers such as concurrent process execution by the operating system (such as context switching), which could have a performance impact on the application (for example on the cache utilization).

Also, to minimize external process interference and give a more accurate picture of the application performance, all concurrent applications should be closed before running the tests. This measure should keep the context switching at a minimum and give the best results that could be expected from this application.

A set of specific images and CIF files were used for the validation tests. Also, the ROIs should have the same coordinates selection. To achieve this, the application project files were used, which have all this information and also the user's configurations for the image indexation.

The application exports a log file with the results, which contains the execution time for the indexation of each CIF file as well as the overall time duration to index each ROI. These values are used to plot a graphical representation of the execution times at each computing platform.

4.2 TOOL VALIDATION AND RESULTS

This section presents graphs with the average and median execution times of each ROI indexation group, at each computing platform.

Note that the indexation process is executed in parallel, so the total real execution time cannot be summarized to the simple sum of the times of each indexation, since they are the times each processing unit took. This subject will later be discussed in another section dedicated to the analysis of parallelism and vectorization of the application.

4.2.1 *Overall time*

Figure 16 shows the measured execution times in each run at each system. These results considered the elapsed time between the time the user chooses the application project file to be loaded until the indication of the last indexation been performed.

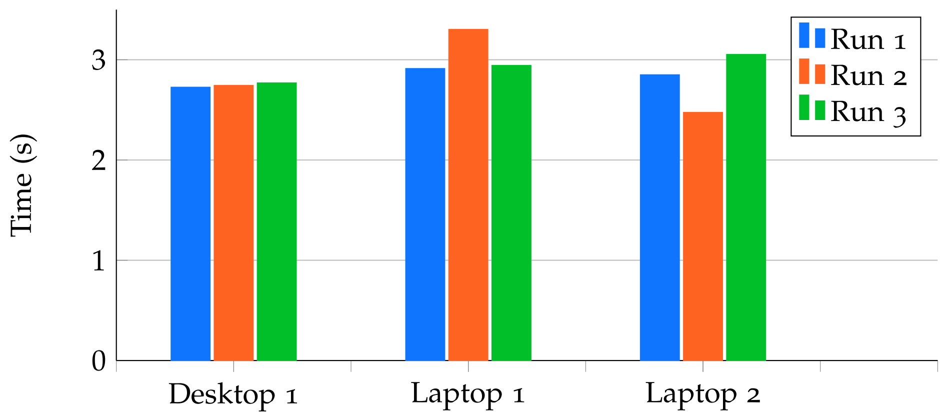


Figure 16.: Project load and indexation time per run

By analysing the values, it is possible to notice a similarity between values in Desktop 1. However, both Laptops 1 and 2 contain times with some discrepancy values. After a more detailed analysis of the results obtained through a larger number of runs, it was possible to conclude a relation between the times obtained with the clock frequency of the CPU at the time of execution of the tests.

This way, due to the oscillating temperatures of the laptops, when the CPU temperatures approached smaller values, its clock speed increases, thus allowing lower execution times. In turn, when temperatures are higher, the top clock speed margin of the CPU is lower, thus recording higher execution times.

This fact will be taken into consideration for a later analysis of the execution times recorded at each system.

However, based on the graph of figure 16, it is possible to conclude that the application execution times are in the interval between 2.5 and 3.5 seconds. It is also visible that the best recorded time is for the most recent laptop, with higher level of parallelism and clock speed when compared to the other systems.

4.2.2 *Desktop 1*

Analysing the application performance on the desktop 1 in figure 17, it is possible to clearly visualize a variation in execution times depending on the ROI to be indexed. This behaviour is normal and is expected, since in these tests a set of images with a set of CIF files were joined without taking into account the real relevance between both. In this sense, it is expected that the images whose indexation is not relevant to have lower execution times.

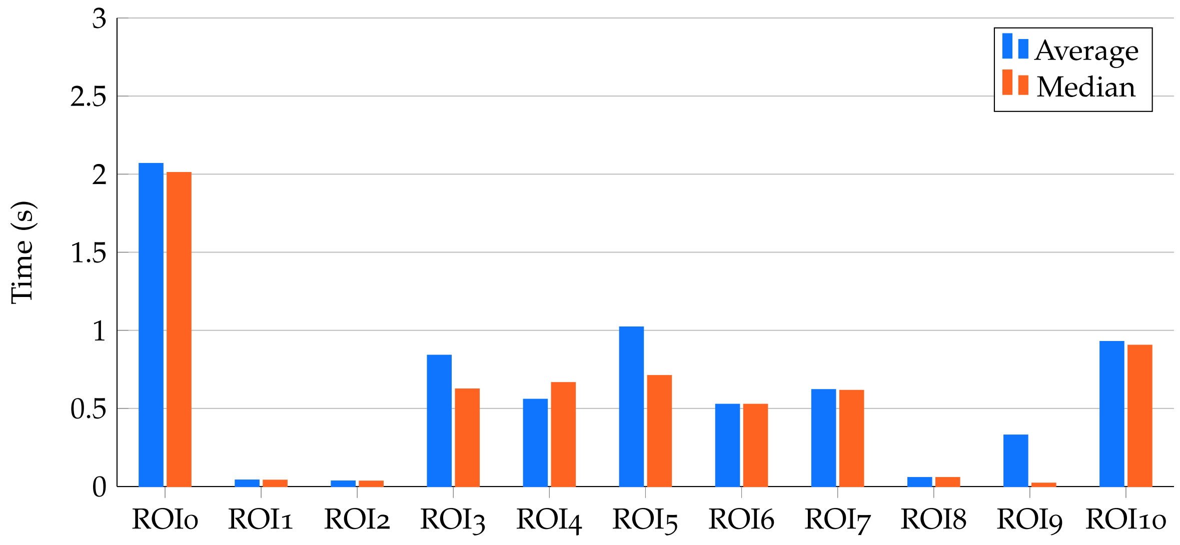


Figure 17.: ROI indexation times for Desktop 1

From the plotted execution it is possible to conclude that with the exception of the ROI with the longest execution time, all other indexes had times lower than 1 second. Similarly, the four shortest indexation execution times are less than 100 ms long.

When analysing the execution times of the three runs, the most significant ones, some occurrences had two runs with similar duration and a third run with an unexpected result. This can be visualised at the median column: when the median bar is considerably shorter than the average one, is a clear indication that one run took considerably longer than expected. In Desktop 1 this happened with ROI9, where the value of two runs were much lower than the time recorded in the third run.

The reason for this discrepancy can be attributed to the way in which indexing is performed, that is, the execution of these tests is carried out in parallel, so discrepancies in time between tests are expected, but it is expected that the total time of the indexation process to be similar or very close to the total loading and indexing time of the project, as shown in figure 16.

In this system, adding all execution times and dividing the value by the number of CPU threads results in a theoretical execution time per thread. In this exercise, the sum of average times is 6995.18 ms, which when divided by 4 CPU threads results in a total indexation time per thread of 1748.8 ms.

This way, it is possible to have an idea of the time dedicated only to the indexation process without taking into account the time dedicated to the other processes involved in loading and presenting information from a file loaded from disk.

4.2.3 *Laptop 1*

The test results performed in Laptop 1 are represented in figure 18.

When comparing these values with those presented in figure 17, previously analysed, it is possible to notice a pattern with respect to the execution times by ROI. In these results the comparison between the average time and the median shows very similar values, which reveals low variation between runs.

However, in terms of absolute execution times per ROI, these are generally higher than those recorded in Desktop 1. This variation is normal and expected since the processors have different clock frequencies and similar architectures. Likewise, the heat dissipation capacity is different, with Desktop 1 theoretically superior in this aspect, which improves its performance by boosting the clock frequency. On the other hand, the number of threads available in Laptop 1 is twice the number of threads in Desktop 1.

In these results it is worth noting a higher indexation time per ROIs, the increase being approximately doubled, with the exception of the ROI with the highest indexation time, whose time has increased by approximately 500 ms.

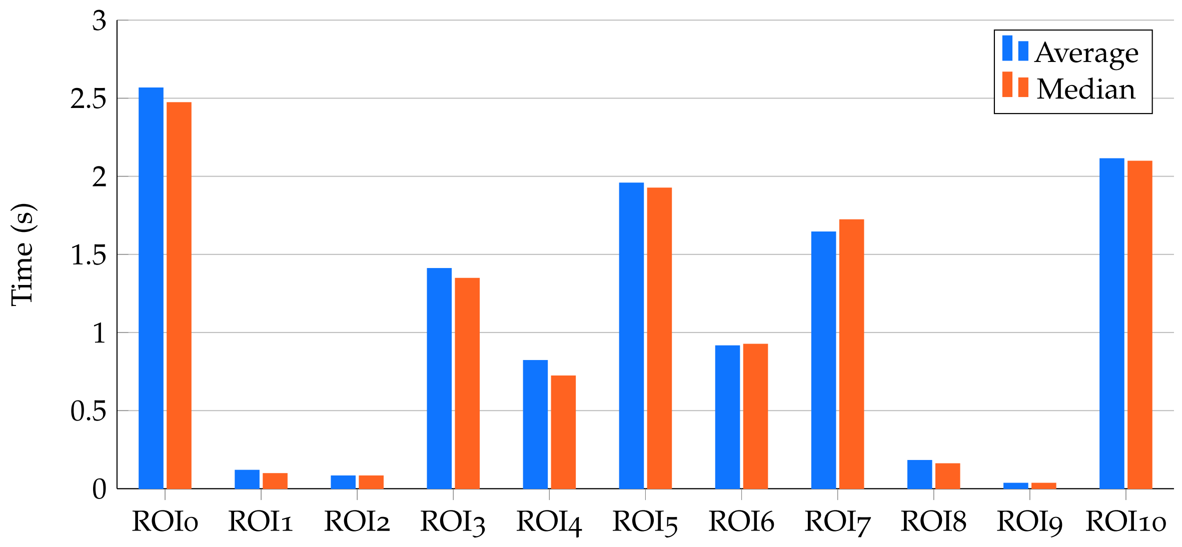


Figure 18.: ROI indexation times for Laptop 1

By performing a theoretical calculation of indexation time per thread it is possible to conclude that the sum of average times totals 11801.66 ms. This value is higher in about 4.8 seconds, representing an increase of 69% when compared with Desktop 1. When dividing this value by the number of threads in the system, the average theoretical ROI indexation time per thread is 1475.21 milliseconds, which is 16% lower than the one registered in Desktop 1.

Comparing the average times of the three tests between machines, Laptop 1 experienced a time increment of 11%, when compared to Desktop 1.

Based on this analysis, it is possible to conclude that, despite of the apparent significant increase of indexing time by ROI, the existence of a greater number of threads available causes the application to maintain execution times similar to those registered in a machine with higher performance per core, thus making the parallelism capability of the application a significant aspect to be taken into consideration. This point will be discussed in more detail in the next section.

4.2.4 *Laptop 2*

Following the previous analysis methodology, the same tests were performed on a more recent laptop. The execution times are plotted in figure 19. To be noted that these results follow the same pattern obtained in figure 18.

In terms of variations, this graph shows some significant variations, namely in ROIs 1, 2, 7 and 9. In these ROIs, and with the exception of 4 indexations, the median value is lower

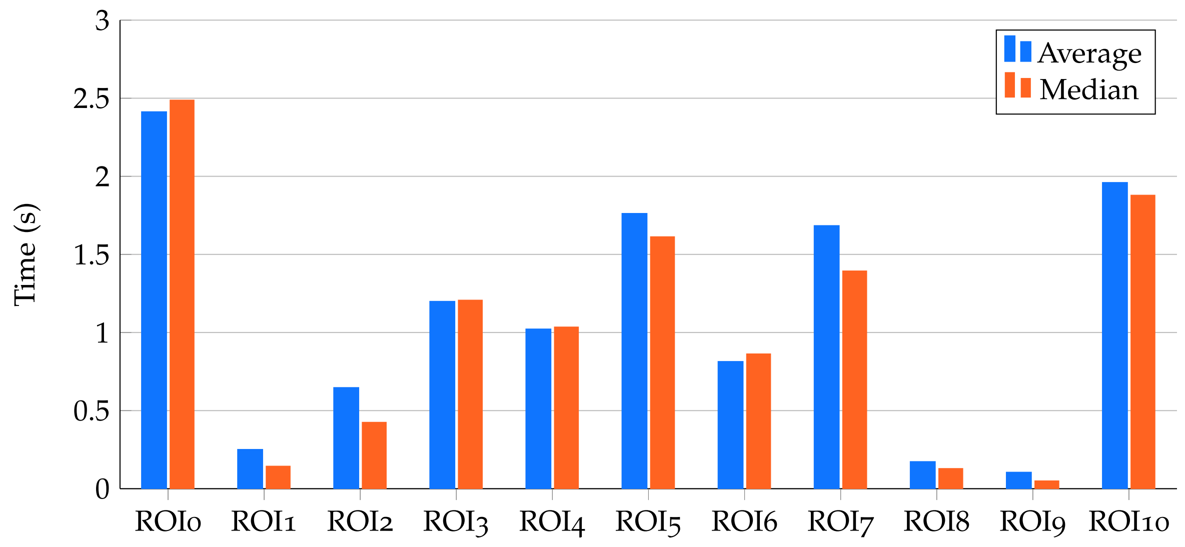


Figure 19.: ROI indexation times for Laptop 2

than the computed average, which means that the value of the third largest execution time was significantly higher to translate into a visible difference on the graph.

The difference between measured values can be considered normal if taken into account the parallel execution of the application. This way, the set of ROIs to be indexed at any given moment is arbitrary and managed by the processor scheduler. Similarly, the concurrent execution of the indexes can contribute to different times per execution, but it is expected that the time to index all ROIs to remain similar. In this aspect figure 16 illustrates that the variation between tests totals about 500 ms.

The sum of the average times totals 11959.55 ms. This value divided by the number of threads available on the system gives an average of 1494.94 ms per thread. Both values are slightly above that obtained in Laptop 1. However, Laptop 2 has a better execution time of about 500 ms, lower than the best time registered by Laptop 1; therefore, a potential of better application performance is acknowledged for Laptop 2.

In relative terms, the sum of the indexation times of Laptop 2 are 71% higher than the ones registered in Desktop 1 and 1.3% higher than Laptop 1. The average time per thread of Laptop 2 is 15% lower than Desktop 1 and 1.3% higher than Laptop 1.

4.3 Vectorization and code parallelism

The previous results were achieved extracting the most performance possible from the computer system. This was possible by using all available threads and by parallelizing computer instructions using vectorization.

To graphically visualise the performance of the application the extension *"Concurrency Visualizer for Visual Studio"* was used. This tool monitors several performance indicators with the objective of analysing the use of parallelism in modern processor architectures.

Figure 20 shows a simple execution of the application to determine the zone axis and upward vector from an image of an unknown material.

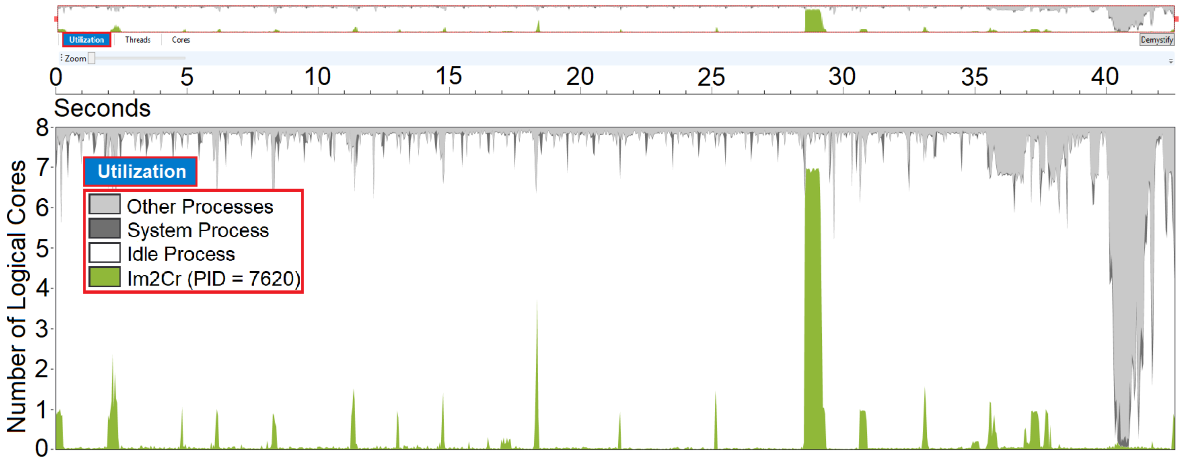


Figure 20.: Performance of complete process for one image and multiple CIF files

The profiled execution makes use of one image and thirteen CIF files. This information delimits a ROI to be indexed and exports the indexation results into an HTML report file.

Looking at the graphical representation of the execution times it is possible to visualise that the system where it was executed had 8 available threads. In the vast majority of the time the application had near the totality of the computational power available. This can be seen through the white colour in the graph.

Represented with green colour is the effective use of the application of the available threads. In terms of runtime, the application was active for about 45 seconds.

Performing an analysis of the data represented, it is possible to see a peak with a longer duration at around the 27 s mark with an approximate duration of 1 s. In this peak it is possible to observe also that the application used 7 of the 8 available threads. This peak represents the indexation process of a ROI in which the application makes use of the parallelism available in the multicore device, with the exception of the last thread.

The other peaks in the image do not have a significant duration.

The main objective of figure 20 is to illustrate the performance of the application using the graphical user interface for interaction with the user. In this case it is clearly visible a bottleneck of the human factor.

In order to illustrate the actual performance of the application, it is important to remove the human factor from the performance analysis. However, since it is an application with a graphical user interface and with the aim of being used by a person, it is important to demonstrate the scenario where its performance may be relevant in the time taken for image analysis.

To illustrate a more intensive use scenario, the application performance was analysed with the data used to obtain the results analysed in the previous section. In this scenario, after starting the application, an application project file with all the information necessary for indexing was selected. The results are shown in figure 21.

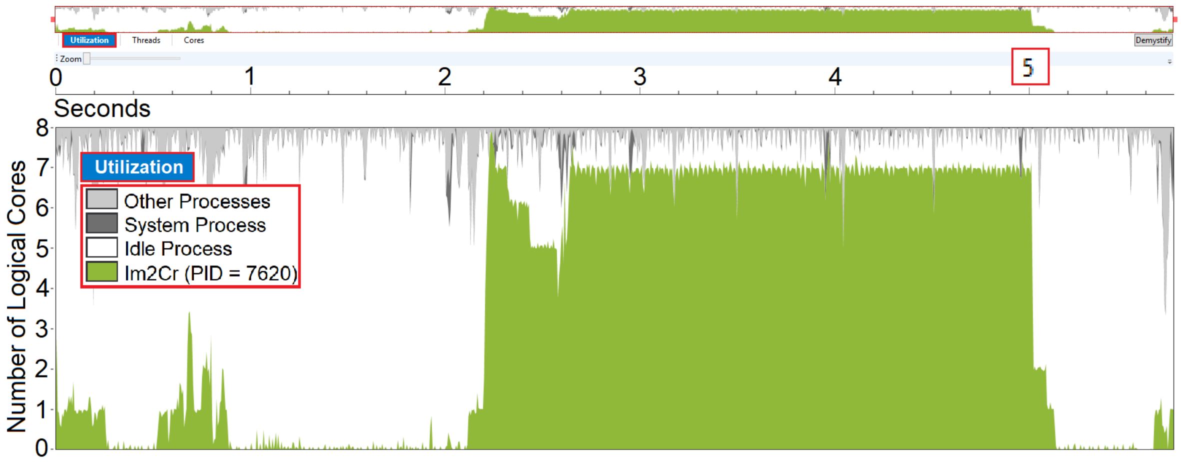


Figure 21.: Performance for indexing multiple images and multiple CIF files from a project file

Similar to the previous figure, also in this graph are present the 8 available threads in the vertical axis and the elapsed time in the horizontal axis. With respect to the time axis, in this test the opening and indexing of an application project file takes about 5.8 seconds with the indexation of 11 ROIs obtained from 7 images and related to 12 CIFs.

Regarding the represented elapsed times, it can be divided into different phases:

1. initialization of the application with the graphical interface: from 0 s to about 0.25 s mark;

2. inactivity by user reaction time: from 0.25 s to 0.5 s mark, followed by execution of the project loading interface up to 1 s;

3. selection of the project file in the GUI: from 1 s to 2 s mark;

4. start loading images, CIFs, configurations and ROIs, followed automatically by the indexation of the meanwhile loaded ROIs: from 2 s to 5 s mark;

5. inactivity due to the human reaction time and closing action of the application in the final section of the graph: the remaining time.

Based on the time axis of the graph it is possible to conclude that the application ran for about 3 seconds on Laptop 2, which is in line to the results presented in the previous section. It is also possible to observe that the application makes use of 7 of the 8 available threads during most of the time during this procedure. This is due to the use of a thread pool functionality provided by the Intel Threading Building Blocks library. In this particular implementation it is in the interest of the application to maintain an active thread available for control of the GUI. For this reason, the application always maintains a thread allocated to the GUI and in which the user can continue to use the tool without interfering in the indexation process that occurs in the background.

Similar to the analysis performed for a normal execution of the application, in this figure it is possible to verify the large amount of computational power available that is not used. For the most part this unused potential result from the graphical implementation of the tool, on which there is no way of increasing efficiency without modifying the library used.

Likewise, it should be mentioned that the displayed results are from the compilation of the application making use of the automatic optimisation from the compiler that favours performance over file size. These optimizations range from the automatic placement of inline functions, to reduce function calls and associated stack accesses, through analysis and pre-allocation of registers for the most used variables, optimisation of loops by extracting static computations, replacing them with variables, replacing some functions by optimised versions present in the processor architecture, releasing and making use of the EBP register, among others.

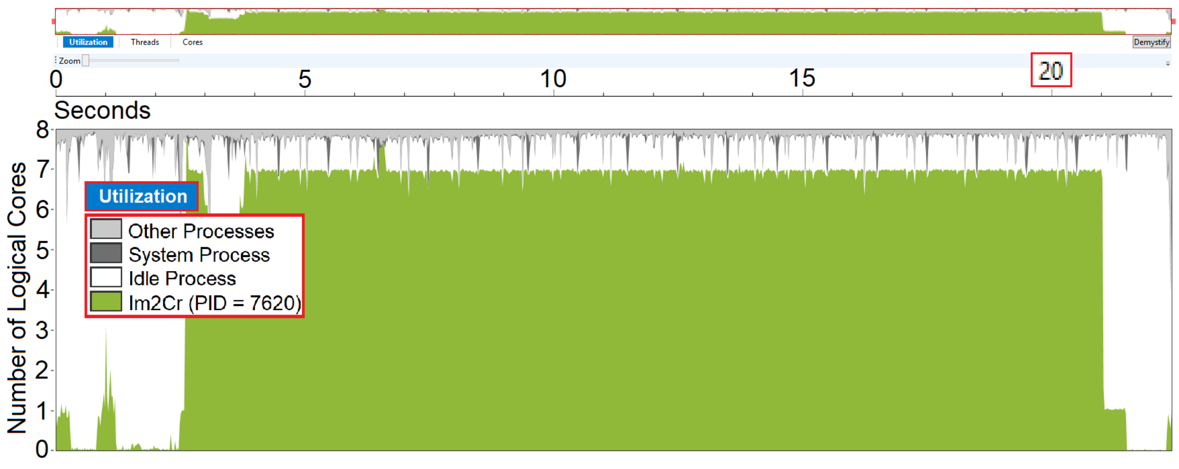


Figure 22.: Same test as figure 21 but with a non optimised executable

Also, the application build makes use of AVX vectorization, whose support was started in 2011 with Intel's Sandy Bridge and AMD's Bulldozer architectures. This feature allows the simultaneous computation of vector instructions on data equal or larger than 128 bits.

To illustrate the lack of optimisation, a test version was compiled without any optimisation and performed the same test of figure 21. The results are shown in figure 22.

As it can be seen in the figure, not even maintaining the parallelism of the application can compensate for the significant penalty in terms of execution time during the indexation process. The indexation process goes from about 3 s of the optimised version to about 19 seconds, 6.3 times slower.

5 CONCLUSION

2D (S)TEM images of nanomaterials may require image interpretation to adequately characterise the atomic structure of the material(s) in the image, in a process known as image indexation. Some optimised approaches do exist to perform some of the required tasks, but there is not yet a fully integrated software tool to perform these tasks in almost real-time, with a simple and efficient user interface. A new tool was developed in this work to address the indexation process, Im2Cr.

The work described in this dissertation shows that this process may be simplified allowing EM users to have a less steep learning path to extract useful information from crystallographic EM images. This does not restrict experienced users from adjusting the tool's parameters to achieve clearer indexation results. For both of them, the advantages are the less time it takes to identify the sample orientation and the better use of their time to do more useful work.

To allow the user to understand the indexation results, Im2Cr makes use of 3D models representation of the unit cell in the orientation found and a value for each indexation representing its classification. With this later value, the user is capable of deciding if the indexation result is trustworthy or not.

With relation to the tolerances allowed by the tool, this application is capable of reading popular image format files and accepting a wide range of images, being them TEM or STEM images with some distortion or not. This flexibility was achieved by allowing the user to perform refinements on the automated spot location while selecting each ROI location.

Taking into consideration the end user requirements, it can be said that most of the requirements were achieved, being this a semi-autonomous tool that can perform image indexation requiring only the user to input the images, CIFs and selecting the ROI area. From this information the tool computes the Fourier Transform of each ROI and indexes the detected spots to each CIF, displaying at the end a table with a orientation and quantitative evaluation of each indexed material. All this is performed on a common modern laptop with a broadly used operating system and with low execution times, when compared with the image acquisition time.

Regarding the application performance, this tool strives to hit a balance point between ease of use and efficient computational power utilisation. As the results shows, the use of a GUI favours an intuitive interface in respect to the used computational power. To counterbalance this, the application tries to perform most computations at the earliest moment when all information is gathered. An example of this is when the user finishes selecting a ROI that, even if the user wants to continue selecting more ROIs, the application starts performing the indexation process on background, allowing the results to be displayed as quickly as possible.

The key components for the application performance are the implemented and/or enabled optimisations. This fact is also emphasised on the displayed results, being the performance of the optimised application compared with an unoptimized version of the same tool. The gains are clear, being the tool 6.3 times faster on the optimised version.

As a conclusion, this dissertation aimed to show that the automation of a routine process as image indexation is possible and that it is an actual problem with many ways to be solved. The developed tool is intended as a contribution to solve this problem, proposing a new approach for its automation with reliable results. Looking at the results, the times achieved are on pair or better when compared with other approaches and are a significant step forward compared to the time taken to perform this process manually.

5.1 FUTURE WORK

At its current state, Im2Cr still have some room for improvements. One of such improvements is the support of diffraction pattern images.

Since the applications allows the detection of spots on the reciprocal space, the implementation of electron diffraction should be straightforward. The main difference for the (S)TEM images is that the process of applying the FT to the image is no longer required since the image is already on the reciprocal space. Instead, the challenge is the determination of spacings by ring radius on the images. The representation of a beam stopper could also have impact on a possible automation of ring detection.

Another point of expansion could be the integration of this tool into the software used for image acquisition. With such functionality, the user could save a significant amount of time when trying to capture a specific orientation of the sample.

The integration on other tools could also be a path to be taken. As an example, image indexation is a requirement for image simulation process. Its current state could be integrated into an application specialized in image simulation, having the indexation results exported by this application being used as input for the image simulation tool.

While developing the tool, a special care was taken in keeping the inner components into modules. The main objective for this was to keep the potential to use this tool on a server. This would allow to have multiple remote clients where the user only performs the required inputs. All the heavy work would be done on the server, where the resources could be more efficiently used.

With relation to known limitations, the current state only allows the representation of orthogonal unit cells. The parameterisation of the tool is not complete yet, having some trouble handling arbitrary user inputs. The background indexation process still requires some adjustment to abort currently running processes.

All in all, Im2Cr have many ways to grow and either be a fully-fledged application or being integrated into other applications. The above mentioned is only a vision of this tool potential. The path this tool will take is still open for future developments.

BIBLIOGRAPHY

H. J. Bunge. *Texture Analysis in Materials Science*. Butterworth, 1982.

Rauch E. and Dupuy Laurent. Rapid diffraction patterns identification through template matching. *Archives of Metallurgy and Materials*, 50:87–99, 2005. URL https://www.researchgate.net/publication/279887412_Rapid_Diffraction_Patterns_identification_through_template_matching.

Rauch E. and Dupuy Laurent. Comments on 'on the reliability of fully automatic indexing of electron diffraction patterns obtained in a transmission electron microscope'. *Applied Crystallography*, 39:104–105, 2006. URL http://doi.org/10.1107/S0021889805033157.

E. J. Groth. A pattern-matching algorithm for two-dimensional coordinate lists. *Astronomical Journal*, 91:1244–1248, 1986. URL http://doi.org/10.1086/114099.

Intel. Intel guide for developing multithreaded applications. *Intel Modern Code Developer Community*, 2012. URL https://software.intel.com/en-us/articles/intel-guide-for-developing-multithreaded-applications.

C. L. Jia, S. B. Mi, J. Barthel, D. W. Wang, R. E. Dunin-Borkowski, K. W. Urban, and A. Thust. Determination of the 3d shape of a nanoscale crystal with atomic resolution from a single image. *Nature Materials*, 13:1044–1049, 2014. URL http://dx.doi.org/10.1038/nmat4087.

M. Klinger and A. Jäger. Crystallographic tool box (crystbox): automated tools for transmission electron microscopists and crystallographers. *Applied Crystallography*, 48:2012–2018, 2015. URL http://doi.org/10.1107/S1600576715017252.

Yifei Meng and Jian-Min. Zuo. Improvements in electron diffraction pattern automatic indexing algorithms. *The European Physical Journal Applied Physics*, 80, 2017. URL http://doi.org/10.1051/epjap/2017160444.

A. Morawiec and E. Bouzy. On the reliability of fully automatic indexing of electron diffraction patterns obtained in a transmission electron microscope. *Applied Crystallography*, 39: 101–103, 2006. URL http://doi.org/10.1107/S0021889805032966.

A. Silva and H. Sobral. An efficient unattended peakfinder for hrstem images. *Project Report at Parallel & Distributed Computing major courses in Master Degree in Informatics Engineering, University of Minho*, 2015.

A. Silva, E. Carbó-Argibay, A. Proença, and D. Stroppa. Im2cr: An efficient tool for crystallographic indexing of hr(s)tem images (poster). *European Microscopy Congress 2016: Proceedings*, pages 585–586, 2016. URL http://doi.org/10.1002/9783527808465.EMC2016.6803.

Daniel G. Stroppa, Ricardo D. Righetto, Luciano A. Montoro, Lothar Houben, Juri Barthe, Marco A. L. Cordeiro, Edson R. Leite, Weihao Weng, Christopher J. Kiely, and Antonio J. Ramirez. Assessment of a nanocrystal 3-d morphology by the analysis of single haadf-hrstem images. *Nanoscale Research Letters*, 8:475, 2013. URL http://dx.doi.org/10.1186/1556-276X-8-475.

Stefan Zaefferer. New developments of computer-aided crystallographic analysis in transmission electron microscopy. *Applied Crystallography*, 33:10–25, 2000. URL http://doi.org/10.1107/S0021889899010894.

A

SUPPORT MATERIAL

Im2Cr: An efficient tool for crystallographic indexing of (S)TEM images and ED patterns

Universidade do Minho
Escola de Engenharia

André Silva[1], Enrique Carbó-Argibay[2], Alberto Proença[1], Daniel Stroppa[2,3]

1 – Algoritmi Centre, University of Minho, Braga, Portugal; 2 – International Iberian Nanotechnology Laboratory, Braga, Portugal; 3 – FEI Company, Eindhoven, The Netherlands;

andresilvaptmail@gmail.com ***daniel.stroppa@fei.com***

INTERNATIONAL IBERIAN NANOTECHNOLOGY LABORATORY

This work presents Im2Cr, a new software tool to aid the crystallographic indexing of nanostructured materials using high resolution (S)TEM images and electron diffraction patterns. Im2Cr implementation aims for a minimal user interaction and includes an efficient peak detection process applied to images and/or their Fourier Transform (FT). Crystallographic indexation is carried out autonomously via comparison with a list of candidate structures named by the user, and a ranking of the best matching combinations of crystallographic structures and viewing zone axes is generated. Im2Cr was successfully tested for robustness and execution efficiency in a wide range of (S)TEM images from crystalline nanomaterials, with domain size ranging from 4 to 100 nm. The autonomous indexation with preset parameters has a very high success rate, and runs in a small fraction of typical (S)TEM images acquisition time. Alternatively, the user can operate Im2Cr in a semi-autonomous mode and control relevant parameters related to the region of interest (ROI) selection on the (S)TEM image and on the peaks detection. Im2Cr promising results point to the possibility of real-time TEM results analysis with reduced user interaction, allowing for an increased (S)TEM characterization yield and also enabling the interpretation of complex images and diffraction patterns.

Motivation

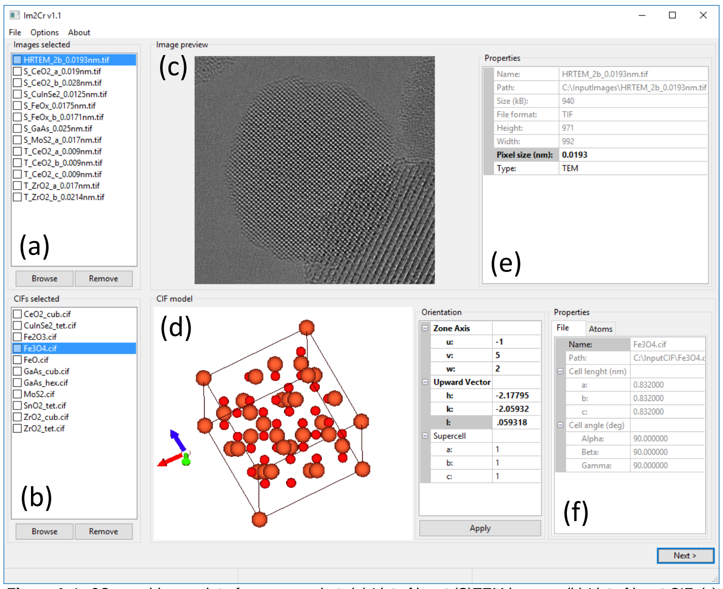
- (S)TEM image and ED patterns indexation is typically done manually and prone to user mistakes
- Real-time TEM results indexation would support the characterization of complex materials
- TEM characterization yield could be greatly improved by (semi-) automated data analysis

Im2Cr functionalities

- Support for popular (S)TEM image formats – TIFF, DM3 and SER
- Flexible 3D visualization of unit cell structures from Crystallographic Information Files (CIF)
- Simultaneous processing of multiple images, diffraction patterns and CIFs
- Straightforward customization of peak finding algorithm and spots indexing
- Time-efficient images and ED indexation – (<1 s)
- Automated report routine that includes indexing results and source images

Input information

- Multiple (S)TEM images and/or ED patterns with pixel size information
- Multiple CIFs including atomic coordinates



Images selected: HRTEM_2b_0.0193nm.tif, S_CeO2_a_0.019nm.tif, S_CeO2_b_0.028nm.tif, S_CuInSe2_0.0125nm.tif, S_FeOx_0.0175nm.tif, S_FeOx_b_0.0171nm.tif, S_GaAs_0.025nm.tif, S_MoS2_a_0.017nm.tif, T_CeO2_a_0.009nm.tif, T_CeO2_b_0.009nm.tif, T_CeO2_c_0.009nm.tif, T_ZrO2_a_0.017nm.tif, T_ZrO2_b_0.0214nm.tif

Image preview

Properties

Name:	HRTEM_2b_0.0193nm.tif
Path:	C:\InputImages\HRTEM_2b_0.0193nm.tif
Size (kB):	940
File format:	TIF
Height:	971
Width:	992
Pixel size (nm):	**0.0193**
Type:	TEM

CIFs selected: CeO2_cub.cif, CuInSe2_tet.cif, Fe2O3.cif, Fe3O4.cif, FeO.cif, GaAs_cub.cif, GaAs_hex.cif, MoS2.cif, SnO2_tet.cif, ZrO2_cub.cif, ZrO2_tet.cif

CIF model

Orientation

Zone Axis	
u:	-1
v:	5
w:	2
Upward Vector	
h:	-2.17795
k:	-2.05932
l:	.059318
Supercell	
a:	1
b:	1
c:	1

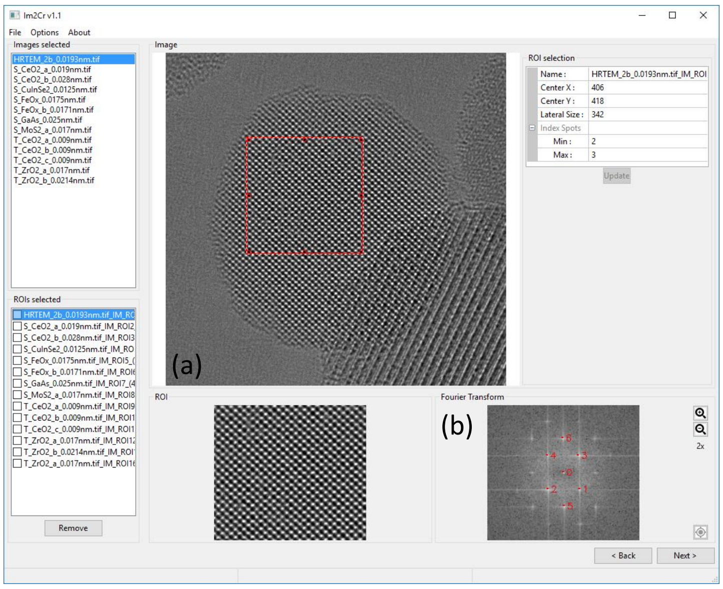
Properties (File | Atoms)

Name:	Fe3O4.cif
Path:	C:\InputCIF\Fe3O4.c
Cell lenght (nm)	
a:	0.832000
b:	0.832000
c:	0.832000
Cell angle (deg)	
Alpha:	90.000000
Beta:	90.000000
Gamma:	90.000000

Figure 1: Im2Cr graphic user interface screenshot. (a) List of input (S)TEM images, (b) List of input CIF, (c) (S)TEM image preview, (d) Unit cell structure preview, (e) (S)TEM image details, (f) Unit cell structure details.

ROI selection and Centre determination

- One or more regions of interest (ROI) may be selected in each of the (S)TEM images input
- ED patterns centre determination is carried out after an estimation by the user
- Automatically detected spots on the ROIs Fourier Transform (FT) and ED patterns can be edited
- The number of indexed spots is customizable



Images selected: HRTEM_2b_0.0193nm.tif, S_CeO2_a_0.019nm.tif, S_CeO2_b_0.028nm.tif, S_CuInSe2_0.0125nm.tif, S_FeOx_0.0175nm.tif, S_FeOx_b_0.0171nm.tif, S_GaAs_0.025nm.tif, S_MoS2_a_0.017nm.tif, T_CeO2_a_0.009nm.tif, T_CeO2_b_0.009nm.tif, T_CeO2_c_0.009nm.tif, T_ZrO2_a_0.017nm.tif, T_ZrO2_b_0.0214nm.tif

ROIs selected: HRTEM_2b_0.0193nm.tif_IM_RO, S_CeO2_a_0.019nm.tif_IM_ROI2, S_CeO2_b_0.028nm.tif_IM_ROI3, S_CuInSe2_0.0125nm.tif_IM_RO, S_FeOx_0.0175nm.tif_IM_ROI5_(, S_FeOx_b_0.0171nm.tif_IM_ROI6, S_GaAs_0.025nm.tif_IM_ROI7_(4, S_MoS2_a_0.017nm.tif_IM_ROI8, T_CeO2_a_0.009nm.tif_IM_ROI9, T_CeO2_b_0.009nm.tif_IM_ROI1, T_CeO2_c_0.009nm.tif_IM_ROI1, T_ZrO2_a_0.017nm.tif_IM_ROI12, T_ZrO2_b_0.0214nm.tif_IM_ROI, T_ZrO2_a_0.017nm.tif_IM_ROI16

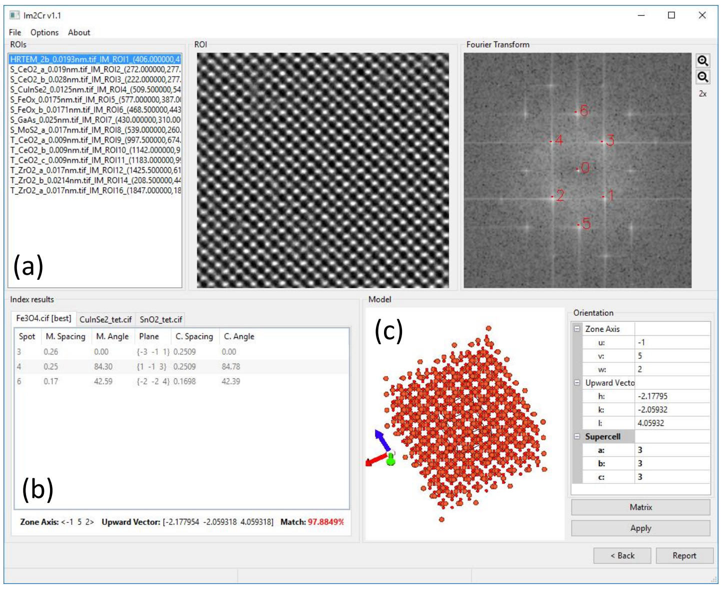
ROI selection

Name :	HRTEM_2b_0.0193nm.tif_IM_ROI
Center X :	406
Center Y :	418
Lateral Size :	342
Index Spots	
Min :	2
Max :	3

Figure 2: ROI selection graphic user interface . (a) The inset (red square) indicates the selected ROI. In case of ED pattern, a centre estimation routine is available instead. (b) An automatic spot detection is displayed for user validation before indexing procedure

Indexation results

- Detected spots are labelled with overlays on each of selected ROI FTs and ED patterns
- The best matching CIF and the indexing results are summarized
- A merit index based on the relative deviation with respect to the CIF structure is displayed
- The crystalline structure orientation is described by the zone axis and upward vectors



ROIs: HRTEM_2b_0.0193nm.tif_IM_ROI1_(406.000000,4, S_CeO2_a_0.019nm.tif_IM_ROI2_(272.000000,277., S_CeO2_b_0.028nm.tif_IM_ROI3_(222.000000,277., S_CuInSe2_0.0125nm.tif_IM_ROI4_(509.500000,54, S_FeOx_0.0175nm.tif_IM_ROI5_(577.000000,387.0, S_FeOx_b_0.0171nm.tif_IM_ROI6_(468.500000,443, S_GaAs_0.025nm.tif_IM_ROI7_(430.000000,310.00, S_MoS2_a_0.017nm.tif_IM_ROI8_(539.000000,260., T_CeO2_a_0.009nm.tif_IM_ROI9_(997.500000,674., T_CeO2_b_0.009nm.tif_IM_ROI10_(1142.000000,9, T_CeO2_c_0.009nm.tif_IM_ROI11_(1183.000000,9, T_ZrO2_a_0.017nm.tif_IM_ROI12_(1425.500000,61, T_ZrO2_b_0.0214nm.tif_IM_ROI14_(208.500000,44, T_ZrO2_a_0.017nm.tif_IM_ROI16_(1847.000000,18

Index results (Fe3O4.cif [best] | CuInSe2_tet.cif | SnO2_tet.cif)

Spot	M. Spacing	M. Angle	Plane	C. Spacing	C. Angle
3	0.26	0.00	{-3 -1 1}	0.2509	0.00
4	0.25	84.30	{1 -1 3}	0.2509	84.78
6	0.17	42.59	{-2 -2 4}	0.1698	42.39

Zone Axis: <-1 5 2> **Upward Vector:** [-2.177954 -2.059318 4.059318] **Match:** 97.8849%

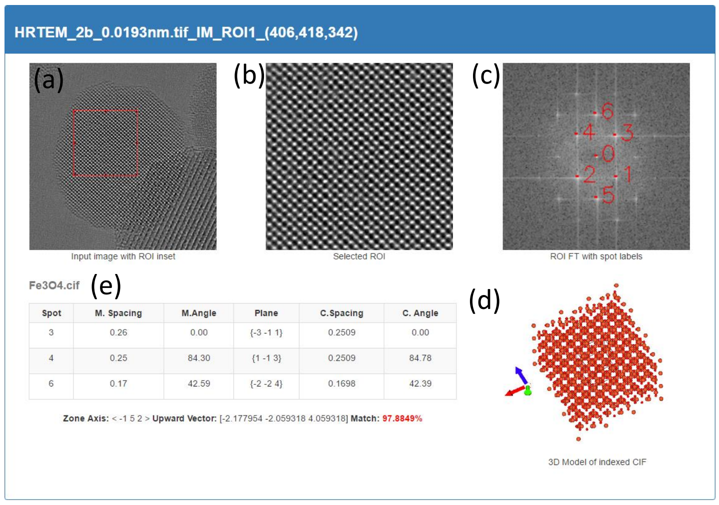
Orientation

Zone Axis	
u:	-1
v:	5
w:	2
Upward Vecto	
h:	-2.17795
k:	-2.05932
l:	4.05932
Supercell	
a:	3
b:	3
c:	3

Figure 3: a) List containing the previously selected ROIs. b) Indexing results for the best matching CIF (Fe_3O_4 - #1011032). c) Interactive 3D atomic model.

Application report

- Indexing results can be exported to a printable HTML file
- Source images and tables can be also exported separately
- Indexing presets and results can be saved in a project file for further application



Fe3O4.cif

Spot	M. Spacing	M.Angle	Plane	C.Spacing	C. Angle
3	0.26	0.00	{-3 -1 1}	0.2509	0.00
4	0.25	84.30	{1 -1 3}	0.2509	84.78
6	0.17	42.59	{-2 -2 4}	0.1698	42.39

Zone Axis: < -1 5 2 > **Upward Vector:** [-2.177954 -2.059318 4.059318] **Match:** 97.8849%

Figure 4: Simplified report example including a) original (S)TEM image, b) selected ROI, c) FT with detected spots, d) unit cell structure, and e) table with indexing results. The individual image can be exported with lossless compression.

Summary

- Im2Cr allows semi-automated and real-time (S)TEM results indexing
- Multiple (S)TEM images, ED patterns and CIF files can be processed simultaneously
- Im2Cr is optimized for parallel processing, allowing fast and scalable analyses for PC
- Quick setup with few-clicks operation, does not require specialized knowledge in crystallography
- Quantitative results and a merit index allows the assessment of indexing quality

Acknowledgements

The authors would like to thank C. Dalmaschio, E. R. Leite, M. Claro, Y. Kolen'ko and M. Bar Sadan for providing numerous samples to test Im2Cr indexing capabilities.