



Article

Int3D: A Data Reduction Software for Single Crystal Neutron Diffraction

Nebil A. Katcho *, Laura Cañadillas-Delgado, Oscar Fabelo , María Teresa Fernández-Díaz and Juan Rodríguez-Carvajal 

Diffraction Group, Institut Laue-Langevin, 71 Avenue des Martyrs, CEDEX 9, F-38054 Grenoble, France; canadillas-delgado@ill.fr (L.C.-D.); fabelo@ill.fr (O.F.); fernandez@ill.fr (M.T.F.-D.); rodriguez-carvajal@ill.fr (J.R.-C.)

* Correspondence: katcho@ill.fr

Abstract: We describe a new software package for the data reduction of single crystal neutron diffraction using large 2D detectors. The software consists of a graphical user interface from which the user can visualize, interact with and process the data. The data reduction is achieved by sequentially executing a series of programs designed for performing the following tasks: peak detection, indexing, refinement of the orientation matrix and motor offsets, and integration. The graphical tools of the software allow visualization of and interaction with the data in two and three dimensions, both in direct and reciprocal spaces. They make it easy to validate the different steps of the data reduction and will be of great help in the treatment of complex problems involving incommensurate structures, twins or diffuse scattering.

Keywords: single crystal diffraction; neutrons; data reduction; software



Citation: Katcho, N.A.; Cañadillas-Delgado, L.; Fabelo, O.; Fernández-Díaz, M.T.; Rodríguez-Carvajal, J. Int3D: A Data Reduction Software for Single Crystal Neutron Diffraction. *Crystals* **2021**, *11*, 897. <https://doi.org/10.3390/cryst11080897>

Academic Editor: Angela Altomare

Received: 5 July 2021

Accepted: 26 July 2021

Published: 31 July 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

X-ray and neutron diffraction are the primary experimental techniques for exploring the atomic structure of condensed matter. They allow deduction with high precision of the electron density, the identity and spatial positions of the atoms, and the magnitude and orientation of their magnetic moments. This knowledge is essential to understand the observed properties of the materials [1], and to test theoretical developments in solid state physics.

The structural information provided by a single crystal diffraction experiment is extracted from the intensities of the indexed diffracted beams [2], using some of the software applications for structure determination or refinement that the crystallographic community has developed over the last decades. This implies that the conclusions drawn from a single crystal diffraction experiment ultimately depend on the accuracy of the integrated intensities.

The data reduction is the process by which the detector records are converted into integrated intensities, and therefore plays a fundamental role in ensuring a correct interpretation of a diffraction experiment. For single crystal diffraction, the subject of this work, the data reduction process must locate the Bragg peaks on the detector, determine the orientation matrix of the crystal, assign Miller indices to the Bragg peaks, integrate the intensity of each reflection and estimate its uncertainty, refine the crystal and instrument parameters, and finally apply absorption corrections to the integrated intensities if required. For X-ray diffraction, there are well established software packages used worldwide for performing all these tasks (CrysAlis [3], APEX3 [4], HKL-3000 [5], DIALS [6]). However, for neutron diffraction the situation is rather different. Although some neutron beamlines have adapted the X-ray software to their needs [7], most of them rely on their own software.

In the particular case of the Institut Laue Langevin (ILL), the situation of the software for single crystal neutron diffraction was far from optimal. The different tasks of the data reduction were not integrated in a single application, and in some cases the codes were

different from one instrument to another. Most of the codes still existing are outdated, written in Fortran77 and designed to work with computers with much less memory than today's computers, which makes them inefficient. They are not ready to work with current data storage like HDF5 and NEXUS files, which makes it necessary to translate these files into ASCII format before they can be processed. They were not designed to deal with superspace symmetry, the most natural way for treating incommensurate crystal and magnetic structures. In addition, finally, they are not coupled to modern graphical libraries offering the possibility to build visualization tools that are of great help to check at run-time, in an easy way, the different steps of the data reduction process.

It is important to stress that all these drawbacks in no way call into question the validity of the algorithms implemented by these codes, which have produced excellent results for 30 years. The reasons for change are to be found in the evolution of both hardware and software, which offer new opportunities for improvement that we must take advantage of. As mentioned above, the field of neutron diffraction has not seen the development of robust software easily adaptable to the different beamlines around the world. It is the purpose of this work to make a contribution to help fill this gap. To this aim, we present a software package, called Int3D, designed to perform the data reduction in single-crystal neutron diffractometers with area detectors. Some parts of the software rely on code previously developed at the ILL. In these cases, either the code has been kept as it was as an interim solution, such as the refinement code, or it has been rewritten in modern Fortran standard, like some parts of the integration code. In most cases, however, the code has been built from scratch. This has been the case of the peak search, the determination of the orientation matrix and the part of the integration related to the generation of reflections, as well as all the graphical environment and the interaction between the graphical tools and crystallographic calculations at run-time.

Int3D is currently used for the data reduction of the data collected at the D19 instrument of the ILL [8]. D19 [9] is a single-crystal diffractometer equipped with an Eulerian cradle and a very large ($120^\circ \times 30^\circ$) horizontal banana-type position-sensitive detector (PSD). We plan to start using Int3D for the data reduction in D9 [10] in the last reactor cycle of 2021. Unlike D19, D9 is also equipped with an Eulerian cradle but with a small PSD that collects reflections one by one. In the near future, Int3D will be extended to D10 [11], a case very similar to D9, and to the new instrument XtremeD [12], a neutron diffractometer for high pressures and magnetic fields with a detector similar to that of D19.

The paper is organized as follows. In Section 2 we give an overview of the Int3D package, describing its software components and the basic operation of its graphical user interface (GUI). Section 3 discusses the main algorithms implemented in Int3D, with special attention to the integration algorithm. In Section 4 we provide an example of a data reduction case, carried out with Int3D, showing the main features of the application. The paper ends with a brief summary highlighting the major achievements and a discussion about future developments. In the supplementary material, we provide a series of short videos showing the different functionalities and graphical tools of the application.

2. The Int3D Package

2.1. Software Components

The structure of the Int3D package is sketched in Figure 1, which shows the software components and how they are interconnected with each other. The component that controls the application is the GUI. It is a Qt application written in Python using PyQt5, a series of Python bindings for the Qt cross-platform C++ framework. Through the GUI, the user can visualize, interact with and process the data. Visualization and interaction are provided by graphical tools that are embedded in the GUI. These tools are built with the Silx [13] and VTK [14] packages. Silx consists of a collection of Python packages that were created to support data assessment, reduction and analysis applications at synchrotron radiation facilities. It provides a set of Qt widgets for data visualization that we have adapted to our needs, in particular for 2D plotting. VTK is open source software for manipulating

and displaying scientific data that provides state-of-the-art tools for 3D rendering. It is implemented in C++ but wrapped with Tcl, Java and Python. We have used its Python wrapper to develop tools for graphical analysis in the reciprocal space. On the other hand, the crystallographic calculations needed for the data processing are carried out by Fortran programs based on the CrysFML [15] library. Some of these calculations are interfaced with the Silx and VTK tools by using Forpy [16], a library for Fortran–Python interoperability that allows writing of Python modules entirely in Fortran. In this way, crystallographic calculations and data visualization are synchronized very efficiently, taking advantage of the high speed of Fortran for numerical calculations and the good visualisation performance of Silx and VTK. The whole package can be distributed as a single folder containing all the required libraries and executables, so that the user can run the application without installing a Python interpreter or any other external module.

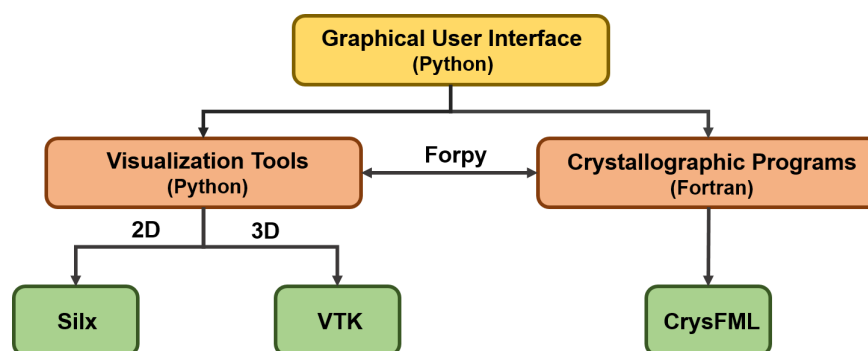


Figure 1. Software components of the Int3D application. See text for details.

2.2. The GUI: An Overview

The GUI is shown in Video S1. It consists of a menu bar, two viewers and a terminal. The menu bar contains four menus: **File**, **Edit**, **Data Reduction** and **Utilities**.

The **File** menu is used to create and load projects. When a project is created, a project directory with subdirectories named ‘Peaks’, ‘Cells’, ‘Integration’ and ‘Refinement’ is created. During the creation of a project, the user must specify the directory where experimental data are located. In a single crystal diffraction experiment, the experimental data are usually constituted by a series of scans. In every scan, the crystal is rotated by a certain angle during the data collection, and the data are stored as a three-dimensional array. Two dimensions refer to the detector and the third one to the rotation angle, so an array element, referred to below as *voxel*, stores the recorded counts at a given pixel of the detector for a given rotation angle. As for large detectors the data of an individual scan can take up several gigabytes in ASCII format, the diffractometers of the ILL have started storing data in files with NEXUS format [17], which have a significant smaller size. Therefore, the experimental input data for Int3D are composed of a series of NEXUS files stored in a single folder.

The **Edit** menu is used to configure the instrument (wavelength, instrument geometry and zero shifts, and detector parameters) and to provide the crystal-dependent parameters (orientation matrix, crystal domains and incommensurate wave-vectors). Instrument and crystal parameters are saved in ‘.geom’ and ‘.cry’ files, respectively, which are stored in the project folder and used by the different programs run by the GUI.

The **Data Reduction** menu contains four actions that allow carrying out of different tasks of the data reduction, namely: the peak search (PeakFind), the determination of the orientation matrix (GetUB), the integration of the intensity of the reflections (Int3D) and the refinement of the instrument and crystal parameters (RefUB). The programs PeakFind, Int3D and RefUB work in a similar way: the GUI displays a window that allows introduction of the required input data for the corresponding Fortran program that will run in the background. Several output files are written in the corresponding running directory (‘Peaks’, ‘Integration’ or ‘Refinement’). These files are described in detail in

Section 4. On the other hand, GetUB, which has its own GUI invoked from the general GUI, is used for performing calculations and analysis in reciprocal space. Its main goal is to assist the user in finding the orientation matrix. Its GUI includes a reciprocal space viewer that allows visualization of the positions of measured reflections in reciprocal space and reciprocal lattices obtained from orientation matrices. The user can interact with the data in a variety of ways. For instance, by measuring the distance between reflections, which is useful for the identification of propagation vectors, by selecting subsets of reflections and classifying them in different groups, which is of great help in case of twinning and by translating all reflections to the same node of the reciprocal lattice, which is useful to check if the instrument offsets are well determined. The selection tool is also useful for defining a subset of reflections for searching the orientation matrix. This allows easy removal of spurious reflections that may hinder the determination of the orientation matrix.

The **Utilities** menu contains actions of different character. Some of them serve as support tools for the data reduction. For instance, a library viewer that allows visualization of the shape of the ellipsoids used for integrating weak reflections, as discussed in Section 3.4.5. Others, however, are not directly related to the data reduction process, such as an ASCII to NEXUS converter for treating old data collected at ILL, and are not covered in the paper.

The left (2D) viewer embedded in the general GUI is a Silx-based widget for displaying the scans in 2D, i.e., it displays the detector counts for a selected rotation angle, labeled as the frame in the widget. The right (3D) viewer is a VTK-based widget for displaying the scans in 3D, i.e., it displays the detector counts for all rotation angles that make up the scan. These viewers were designed to provide a number of graphical tools to assist in the analysis of the collected diffracted intensity and in the validation of the data reduction process. Some useful tasks they perform are: pixel indexing and displaying the calculated peak centroids and the integration ellipsoids, boxes and masks. This is covered in Section 4.

Finally, the terminal is used to display the standard output at run-time of the different Fortran programs that run in the background.

3. Methods

3.1. Peak Search

The peak search consists of scanning the experimental data to identify groups of voxels that make up a reflection. This is performed by the program PeakFind. The search algorithm needs two input values: the threshold and the size (Video S2). The threshold determines the minimum number of counts of a voxel to be considered as part of a reflection. The size imposes a lower limit on the number of voxels that make up a reflection. Using high values for the threshold and the size allows one to search only strong reflections, which is useful for example in incommensurate lattices when only the parent lattice needs to be determined, as explained in Section 3.2. The algorithm starts by identifying all voxels whose counts are above the threshold. These voxels are then grouped together in an iterative process. Two voxels become part of the same group if they are first neighbors. When the iterative process converges, the different groups of voxels that have been found are analyzed. If the number of voxels of the group is greater than or equal to the minimum size specified in the input, the group is considered as a reflection and its center of mass is computed using the intensities of each voxel as a weighting factor. Finally, the coordinates of the center of mass in the direct space are converted into the corresponding scattering vector in the reciprocal space. For the four-circle geometry used in D19, the conversion proceeds in two steps. First, the polar angles γ and ν , determining the direction of the scattered beam, unitary vector \mathbf{S} , in the laboratory system of axes for orienting angles (ω, χ, ϕ) , are obtained. γ and ν refer to the horizontal and out of plane angles, respectively. The laboratory system of axes is set by following the conventions of Busing and Levy [18]. In this frame the incident beam unitary vector has always the components $\mathbf{S}_0 = (0, 1, 0)$.

For an horizontal banana-type detector as that of D19, the transformation is given by the following set of equations:

$$\begin{aligned}x_L &= r \cdot \sin(x_D/r) + x_{off} \\y_L &= r \cdot \cos(x_D/r) + y_{off} \\z_L &= z_D + z_{off} \\\gamma &= \gamma_D + \arctan(x_L, y_L) \\\nu &= \nu_D + \arctan(z_L, \sqrt{x_L^2 + y_L^2})\end{aligned}\quad (1)$$

where subscripts D and L stand for detector and laboratory coordinate systems, respectively, r is the sample-detector distance, and the subscript *off* stands for the detector offsets. x_D and z_D are the detector coordinates of the center of mass of the reflection. γ_D and ν_D refer to the polar coordinates of the detector center. Other relations stand for different kinds of detector geometries.

From the polar angles of the scattered beam \mathbf{S} and the components of the incident beam \mathbf{S}_0 , the calculation of the scattering vector in the laboratory system $\mathbf{h}'_L = \frac{1}{2\pi} \mathbf{q}'$ is straightforward.

$$\mathbf{h}'_L = \frac{\mathbf{S} - \mathbf{S}_0}{\lambda} = \frac{1}{\lambda} (\sin \gamma \cos \nu, \cos \gamma \cos \nu - 1, \sin \nu) \quad (2)$$

where λ is the neutron wavelength. The last step consists of computing the scattering vector \mathbf{h}_L corresponding to all orienting angles set to zero. This is achieved by inverting the following equation:

$$\mathbf{h}'_L = \Omega X \Phi \mathbf{h}_L \quad (3)$$

where Ω , X and Φ are the rotation matrices corresponding to the three orienting angles of the Eulerian cradle. If the scattering vector represents a Bragg reflection of indices $\mathbf{h} = (h, k, l)$ with respect to the reciprocal basis of the crystal, the UB -matrix relates both vectors as: $\mathbf{h}_L = UB \mathbf{h}$, U being the orientation matrix and B the Busing–Levy matrix converting the coordinates of the reciprocal lattice points to a Cartesian system attached to the crystal [18]. The set of reflections \mathbf{h}_L provided by the program PeakFind is the input for the determination of the orientation of the crystal addressed in the next subsection.

3.2. Determination of the Orientation Matrix

The orientation matrix is set by finding the unit cell that best indexes the set of reflections provided by the peak search. The indexing method used by Int3D is based on that proposed by Duisenberg and implemented in the Dirax program [19]. The original algorithm was designed for dealing with quite complex problems as incommensurate lattices, misfits and twins. We have simplified the algorithm somewhat because these complex problems can be handled more easily and safely with the help of graphical tools, absent in the Dirax program. The required input is explained in Section 4. Most of the input parameters are self-explanatory.

The indexing algorithm starts by projecting the reflection scattering vectors on axes that are expected to be oriented along direct lattice vectors. These axes are obtained from a random sampling of the data. In the Dirax program, by computing the normals to randomly chosen triplets of reflections \mathbf{h}_L . In Int3D, by computing first the inter-spot vectors of randomly chosen pairs of \mathbf{h}_L -vectors. If the two reflections of the pair belong to the same reciprocal lattice, the projection should show a regular one-dimensional pattern whose period \mathbf{d}^* is deduced following the iterative procedure proposed by Duisenberg [19]. The period \mathbf{d}^* is the smallest possible vector that indexes the largest number of reflections. The indexing tolerance is set by the parameter called ‘Level Fit’, as in Dirax, that the user can tune from the GUI of GetUB. As the sampling progresses, the \mathbf{d}^* periods that index the highest number of reflections are stored. Before storing a particular \mathbf{d}^* , the program checks that an equivalent vector has not been previously stored. The equivalence between two \mathbf{d}^*

is defined by the tolerances inside the ‘Indexing vectors’ box of the GUI. If an equivalent vector is found, the program keeps the one that indexes best. When the sampling is finished, the best N indexing vectors \mathbf{d}^* , where N is an upper limit set from the GUI, are grouped in triads $\{\mathbf{a}^*, \mathbf{b}^*, \mathbf{c}^*\} = \{\mathbf{d}_1^*, \mathbf{d}_2^*, \mathbf{d}_3^*\}$ and then transformed into their reciprocal counterparts that correspond to direct unit cells. The corresponding UB matrices are obtained with the procedure described in Ref. [18]. The quality of the indexing is quantified with an R-factor. If the R-factor is below the upper limit set in the GUI, the orientation matrix is stored after checking that an equivalent matrix was not previously stored. For all accepted orientation matrices, the Niggli cell is deduced following the algorithm of Krivy and Gruber [20], and then transformed to the conventional cell using the algorithm of Le Page [21], both implemented in CrysFML.

In the case of complex problems where twins or incommensurate lattices are present, the indexing can be quite poor. To solve this problem, Dirax implements an iterative procedure for indexing and refinement. In Int3D the approach is different because of the possibility to interact graphically with the data. In case of twins, the selection tool of the reciprocal space viewer allows the reflections to be classified into different groups, making it easy to obtain sets of reflections for each domain which are then analysed separately. In the case of incommensurate lattices, the propagation vectors can be easily found by determining the orientation matrix of the parent lattice, and then translating all reflections to the origin of this lattice. In this way, the reflections of the parent structure will appear centered at the origin, and the satellite reflections off-center. The magnitude of the propagation vector is easily obtained by looking at the distribution profile along the direction of the incommensurability, as shown in Figure 2.

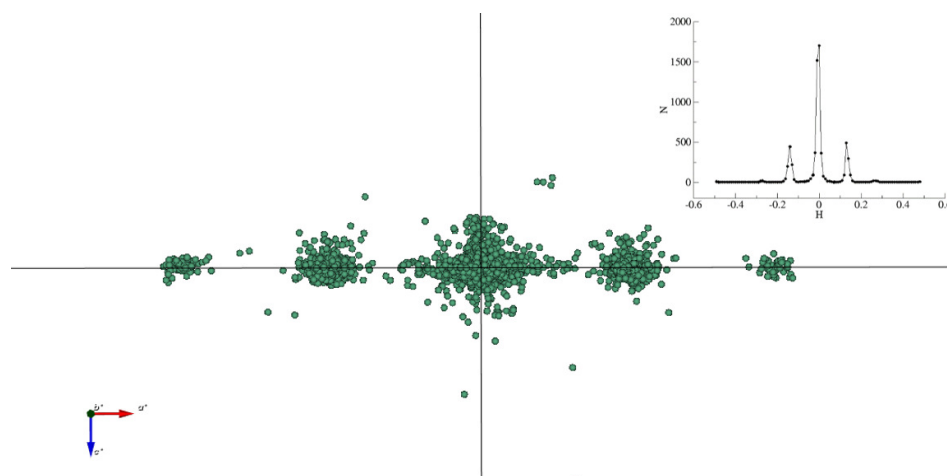


Figure 2. Data collected at D19 for a sample showing incommensurability. Reflections have been translated to the origin of the reciprocal lattice after indexing with the parent cell. First and second order satellites are clearly visible. **Top right:** Distribution profile of the reflections as a function of the h-index.

3.3. Refinement

The quality of the orientation matrix depends on the accuracy of the positions of the reflections in the reciprocal space from which it is obtained. As these positions are obtained from Equations (1)–(3), the orientation matrix depends ultimately on the accuracy of the orienting angles (ω, χ, ϕ) of the Eulerian cradle and the Cartesian coordinates (x_D, z_D) of the pixels detectors, as well as the neutron wavelength λ . The exact value of (ω, χ, ϕ) and (x_D, z_D) depends on the instrument zero shifts, which are usually small but unknown. A suitable method to determine these shifts is that proposed by Busing and Levy [18]. It consists of a non-linear least squares fit that allows refinement of the cell parameters, the cell orientation, the instrument zero shifts and λ . This method was implemented at the ILL in several codes to adapt it to various instrument geometries. Later, the different versions were packed together in a single program called RefUB, which is the one used by Int3D.

The input parameters required by this program for performing the refinement in D19 are classified into two categories: control and refined parameters. ‘Ang Min’ sets the minimum required angle between two reflections to provide the initial orienting parameters. ‘Ang Diff’ sets the maximum acceptable discrepancy between the observed and calculated coordinates. Reflections for which the discrepancy is larger than ‘Ang Diff’ are neglected during the refinement. ‘Max Cycles’ imposes an upper limit to the number of allowed iterations, and ‘Eps’ is the magnitude of the finite increment for the numerical evaluation of the derivative during the minimization. Regarding the refined parameters, an initial value for each of them must be specified. Every parameter has a check box. If it is checked, the parameter is allowed to vary during the refinement. Regarding the lengths and angles of the unit cell, the program allows introduction of internal constraints enabling imposition of the desired symmetry. Finally, an input file containing the information of the observed reflections must be given. This file can be generated either with the peak search or the integration (Section 4).

The reciprocal space viewer can be used to easily check whether the orientation matrix and the instrument offsets are well refined. In Figure 3, the observed reflections are shown before and after refinement. The lines represent the direction axes of the reciprocal lattice corresponding to the refined orientation matrix, and the reflections have been translated to the origin. Figure 3 clearly shows how after the refinement the distribution of reflections around the origin is nearly isotropic, with a smaller dispersion than before the refinement.

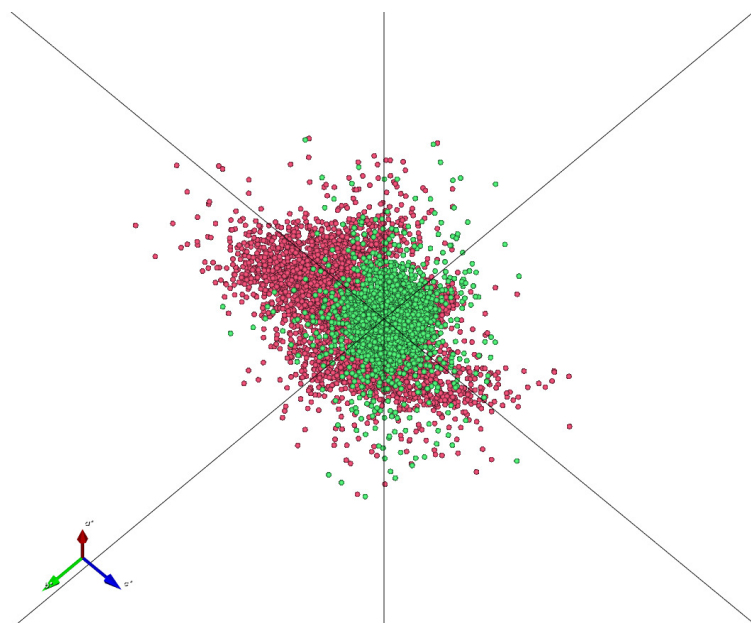


Figure 3. Visual test of the refinement of instrument offsets. Reflections indexed with an orientation matrix before (red dots) and after (green dots) refinement, translated to the origin of the reciprocal lattice (black lines), viewed along [111].

3.4. Integration

The integration of Bragg peaks in Int3D is carried out once an orientation matrix has been set. From the space group symmetry and the orientation matrix, Int3D computes all reflections that are in principle accessible, i.e., those with scattering vector lengths satisfying $\frac{1}{2} |\mathbf{h}| < s_{max} = \sin \theta_{max} / \lambda$, where s_{max} is constrained by the experimental conditions and must be set by the user. The generation of reflections may consider a postulated space group, a magnetic space group or even a superspace group with the corresponding modulation vectors. Then, before the integration of each scan, Int3D runs over the set of accessible reflections and builds a list with those that should appear in the scan. This is done by inverting the calculation described by Equations (1)–(3). Starting from the scattering vector \mathbf{h} , the coordinates x_D and z_D of the reflection in the detector at the scan

conditions are calculated. If x_D and z_D lie within the detector boundaries, the reflection is added to the list.

The integration of the reflections follows closely the algorithm implemented in the program PEAKINT [22], which extends the Lehmann and Larsen $\sigma(I)/I$ integration method [23] from one- to three-dimensional data scans. A version of the program PEAKINT was embedded in the program RETREAT, that was used to perform the data reduction in D19. As this version was written in an old Fortran standard and was designed to work with ASCII files, we have rewritten it completely and adapted it to process NEXUS files and to use the crystallographic procedures implemented in CrysFML, but the integration algorithm remains almost the same. The basic idea of the integration program is to classify reflections into two categories: strong and weak. The shape of the reflection is assumed to be ellipsoidal in the direct space. Strong reflections are used to build a model of the intensity profile, which provides the fraction of the total intensity as a function of the ellipsoid volume. The model is then used during the integration of weak reflections for minimizing the ratio $\sigma(I)/I$. As the intensity profile depends on the scattering angles γ and ν , Int3D defines a grid in the $\gamma - \nu$ space and associates a model to each node of the grid. The node with coordinates (γ_i, ν_j) provides a model profile for a reflection with $\gamma \in [\gamma_i - \Delta\gamma, \gamma_i + \Delta\gamma]$ and $\nu \in [\nu_j - \Delta\nu, \nu_j + \Delta\nu]$. $\Delta\gamma$ and $\Delta\nu$ are set to 1.5° .

The integration algorithm requires passing several times over the voxels to be integrated, and requires several input parameters as shown in Figure 4. In the following we provide a detailed description of the different steps of the integration and their relation to the input parameters accessible to the user. The mathematical justification of the method is well established in Ref. [22].

The screenshot shows the 'int3d' application window. It contains several sections for user input:

- Numors:** A list box containing scan IDs: 139905.nxs, 139906.nxs, 139907.nxs, 139908.nxs, 139909.nxs, 139910.nxs, 139911.nxs, 139912.nxs, and 139913.nxs.
- Crystal cell:**
 - Space group: 1: P 1
 - Unit cell parameters: a (Å) = 8.040778, b (Å) = 8.290357, c (Å) = 11.584586, α (°) = 90.000000, β (°) = 90.000000, γ (°) = 90.000006.
 - UB matrix: A 3x3 grid of values including 0.083212, -0.089638, 0.000752, -0.092267, -0.080645, -0.004399, 0.005434, 0.003334, and -0.086206.
- Integration parameters:**
 - Min/Max for Sin(θ) / λ: 0.0000 / 1.0000
 - Mask: 50.00, No: 5.00, Shell fac: 2.50, BVolt: 500.00
 - Box: ΔX = 40, ΔY = 40, X min = 0, X max = 639, Y min = 0, Y max = 255, Z min = 0, Z max = 0
 - Contours: 0.50 (radio button), 0.20 (radio button), 0.10 (radio button), 0.05 (radio button)
 - Signoi, Psgshp, Monitor, Scan step
 - Strong Test: 1.0000, 5.0000, Normalization: 100000.0000, 0.0700
 - radius_x, radius_y
 - Default integration: 6.0000, 6.0000

Buttons for 'Run' and 'Cancel' are at the bottom right.

Figure 4. Input for integrating the intensity of reflections. The user must select the scans to be integrated, called Numors in the figure. The space group and orientation matrix must have been set in advance, and are read from the .cry file. The meaning of the integration parameters is explained in the text.

3.4.1. Integration Box and Masking

The integration of the reflection intensity is performed inside a rectangular box centered at the center of the reflection, which is computed from the orientation matrix. The size of the box along the two dimensions of the detector is set by the ΔX and ΔY parameters.

A reasonable size can be easily found by visualizing the scans with the Silx-based viewer. The box size along the scan angle is managed internally by Int3D by using a predefined resolution function.

If neighboring reflections are close, it is convenient to apply a mask to some volumes of the box. Int3D looks for reflections that lie inside the integration box, and classifies them as neighbors. For each voxel in the box, Int3D determines its position with respect to the reflection to be integrated, and computes the projections along the lines connecting the reflection to be integrated to each of its neighbors. If at least one of the projections is greater than $m \cdot d/100$, the voxel is masked. m is the input value set in the box labeled as 'Mask' and d the reflection–neighbor distance in voxel units, which is computed internally by the program.

3.4.2. First Pass

The first pass consists of a crude integration with the aim of classifying the reflections as strong or weak. Hereafter we use the notation I and \tilde{I} to refer to the integrated intensity with and without background correction, respectively. The background for every frame is assumed to be flat:

$$B_{1,k} = \frac{1}{V_{B_k}} \sum_{i,j,k \in B} \tilde{I}(i,j,k) \quad (4)$$

where k refers to the frame, i and j refers to a detector pixel, $\tilde{I}(i,j,k)$ stands for the number of counts of the voxel with coordinates i,j and k , the sum runs over all the voxels belonging to the background and $V_{B_{1,k}}$ is the total number of voxels belonging to the background in the frame k . The associated integrated intensity $I_{1,k}$ is then computed as:

$$I_{1,k} = \sum_{i,j,k \in P} [\tilde{I}(i,j,k) - B_{1,k}] \quad (5)$$

where the sum runs now over the voxels belonging to the peak (P). The calculation is performed in an iterative way. Initially, $B_{1,k}$ is obtained by averaging the counts of all the voxels of the frame that have not been masked, and $I_{1,k}$ is set to zero. Then, the iteration starts. A voxel is considered as belonging to the peak if its counts are greater than N times the standard deviation of $B_{1,k}$, where N is set in the box labeled as $N\sigma$ (Figure 4). After each iteration, $B_{1,k}$ and $I_{1,k}$ are updated until convergence is reached. The final integrated intensity I_1 is obtained by summing the $I_{1,k}$ of the frames enclosed by the integration box, whereas the final background B_1 is obtained by averaging the corresponding $B_{1,k}$.

3.4.3. Strong Peak Test

In order to classify a reflection as strong or weak, Int3D performs two tests. In the first test, Int3D evaluates the signal-to-noise ratio:

$$\frac{I_1}{V_1 \cdot B_1} > s \quad (6)$$

where V_1 is the size of the peak in voxels units and s is the input value set in the box labeled as 'Signoi' (Figure 4). The second test checks whether

$$\tilde{I}_{max} \cdot h(1) > p \cdot \sigma(B_1) \quad (7)$$

where \tilde{I}_{max} is the maximum number of counts/voxel of the peak, $h(1)$ the highest fractional height specified in the input (see Section 3.4.4), p the value set in the box labeled as 'Psgshp' (Figure 4) and $\sigma(B_1)$ the standard deviation of B_1 . This second test tries to ensure that a good model can be derived from the reflection. If any of the two tests return false, the peak is considered as weak and the program jumps directly to the fourth pass.

3.4.4. Second Pass

The second pass is used for computing the centroid of the reflection in the direct space and the elements of the tensor ellipsoid at the four different contours specified as fractions of the maximum of the intensity ‘Contours’ item in Figure 4). For each contour C_n , only the voxels (i, j, k) for which $\tilde{I}(i, j, k) \geq \tilde{I}_{max} \cdot h(n)$ enter in the following calculations. The component (coordinate) u of the reflection centroid for contour C_n is:

$$c_u(n) = \frac{1}{\sum_{i,j,k \in C_n} \tilde{I}(i, j, k)} \sum_{i,j,k \in C_n} x_u(i, j, k) \cdot \tilde{I}(i, j, k) \quad (8)$$

$x_u(i, j, k)$ refers to the u th component of the voxel i, j, k , e.g., $(x_1 = i - 1, x_2 = j - 1, x_3 = k - 1)$. The tensor ellipsoid for contour C_n , $\epsilon(n)$, is computed as an inertia tensor assuming that all voxels have the same mass. The components uv of the tensor are then given by:

$$\epsilon_{uv}(n) = \frac{1}{N_n} \sum_{i,j,k \in C_n} \sum_{i',j',k' \in C_n} x_u(i, j, k) \cdot x_v(i', j', k') \quad (9)$$

where N_n is the number of voxels contributing to contour C_n . If Equation (7) returns false for contour C_n , the calculation of the shape for that contour is discarded. The contour selected in the input widget is the contour that will be used by default in subsequent steps if the calculation for that contour succeeds.

3.4.5. Third Pass

The aim of the third pass is two-fold: to obtain the integrated intensity of a strong reflection and to produce the models for the library (see Figure 5). For this purpose, Int3D defines internally a set of volumes V_m that cover the full range of reasonable volumes that a Bragg reflection can have. For every V_m of the set, an ellipsoid E_m with that volume and the shape and centroid found in the second pass is built. The intensity \tilde{I}_m enclosed by the ellipsoid E_m is obtained by summing the counts of all voxels (i, j, k) that lie inside V_m . Assuming Poisson statistics and a flat background, the standard deviation of the integrated intensity I is given by:

$$\sigma(I) = \sqrt{I + V_p B(1 + V_p/V_b)} = \sqrt{\tilde{I} + B V_p^2/V_b} \quad (10)$$

B is the value of the flat background, V_p the number of voxels belonging to the peak and V_b the number of voxels used in the calculation of the background. For strong peaks, the minimum of the ratio $\sigma(I)/I$, i.e., the Lehmann–Larsen criterion, is located close to but at a smaller value than the true boundary of the reflection. Equation (10) can then be used to estimate the peak size by computing the volume at which the ratio $\sigma(I_m)/I_m$ has its minimum, replacing \tilde{I}, B, V_p and V_b with \tilde{I}_m, B_1, V_m and V_1 . To ensure that the reflection is integrated over a volume that encloses the entire reflection, the final integration volume V_{100} is set to:

$$V_{100} = F_{shell} \cdot V_{core} \quad (11)$$

where F_{shell} is the shell factor specified in the input (Figure 4) and V_{core} the volume at which $\sigma(I_m)/I_m$ reaches its minimum value. The background B_3 is then computed by averaging the voxels counts in an outer shell that encloses a number of voxels $V_{B_3} \geq V_{bg}$, where V_{bg} is the background volume in voxel units specified in the input box labeled as ‘BVol’. The final integrated intensity I_s of the strong reflection is obtained after subtracting the background:

$$I_s = I_{100} = \tilde{I}_{100} - B_3 V_{100} \quad (12)$$

$\sigma(I_{100})$ is obtained from I_{100} , B_3 , V_{100} and V_{B_3} and Equation (10).

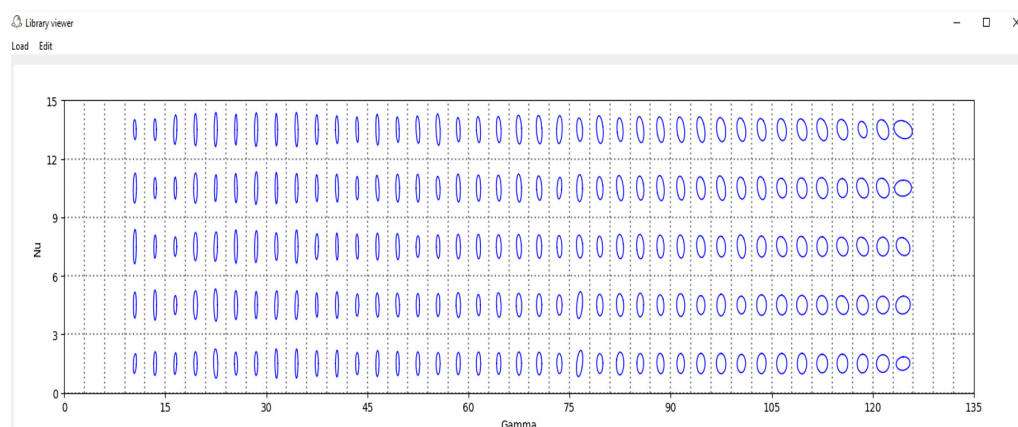


Figure 5. Two dimensional representation of a library (file with ellipsoid parameters) stored by Int3D after integrating data collected at D19. The representation shows the ellipsoid section obtained by computing the intersection between the surface of the ellipsoid and a plane perpendicular to the scan dimension passing through the centroid of the ellipsoid.

Finally, Int3D checks the library slot at the γ and ν values of the current reflection. If the library slot is empty or the model it contains was derived from a weaker reflection, the model for the current reflection is built and inserted into the slot. The model provides the elements of the ellipsoid tensor, which describes the shape and orientation of the ellipsoid, and the fraction of the integrated intensity x_m and its standard deviation $\sigma(x_m)$ for the set of volumes V_m :

$$x_m = \frac{I_m}{I_{100}} \quad (13)$$

$$\sigma(x_m) = \frac{x}{I_{100}} \left[(I_{100} \cdot x + V_m \cdot B_3) \left(\frac{1-x}{x} \right)^2 + (1-x) \cdot I_{100} + (V_{100} - V_m) \cdot B + \left(\frac{V_m}{x} - V_{100} \right)^2 \cdot \frac{B}{V_{B_3}} \right]^{1/2} \quad (14)$$

The library plays a fundamental role in obtaining an accurate integration of weak reflections, discussed in Section 3.4.6. The shape of the ellipsoids can be visualized from the GUI using the ‘Library Viewer’ option in the menu **Utilities**. Figure 5 shows a library almost full of ellipsoids. The loss of resolution as γ increases, that can be well fitted using the Caglioti formula, is clearly visible.

3.4.6. Fourth Pass

The fourth pass is needed to integrate the weak reflections. Initially, Int3D looks at the library to check whether a model is available. If the (γ_i, ν_j) slot to which the weak reflection belongs is occupied (Figure 5), the model from that slot is taken. Otherwise, Int3D checks whether a model can be built by interpolation from neighboring occupied slots. The interpolation is feasible if the occupied slots follow any of the following patterns:

$$\begin{aligned} &(\gamma_{i-n}, \nu_j), (\gamma_{i+n}, \nu_j) \\ &(\gamma_{i-n}, \nu_{j-n}), (\gamma_{i+n}, \nu_{j-n}), (\gamma_i, \nu_{j+n}) \\ &(\gamma_{i-n}, \nu_j), (\gamma_i, \nu_{j-n}), (\gamma_{i+n}, \nu_{j+n}) \end{aligned}$$

where n is allowed to be 1 or 2 (first or second neighbors).

If a model is available, the integration proceeds as follows. First, the ratio $\frac{\sigma^2(I/x)}{(I/x)^2}$, where x is a fraction (see below), is computed for the set of volumes V_m introduced in Section 3.4.5:

$$\frac{\sigma^2(I/x)}{(I/x)^2} = \frac{\sigma^2(I)}{I^2} + \frac{\sigma^2(x)}{x^2} \quad (15)$$

The term $\frac{\sigma^2(x)}{x^2}$ is computed from the model (Equations (13) and (14)). For computing $\frac{\sigma^2(I)}{I^2}$, Equation (10) is used. For each V_m , I_m is obtained as $x_m \cdot I_1$, where x_m is the fraction of the intensity enclosed by V_m , and I_1 is the integrated intensity of the weak reflection computed during the first pass. Then, Int3D determines the x_m and V_m at which the ratio given by Equation (15) is minimized. This volume is taken as the volume V_{core} introduced in the third pass. The volumes called shell and background are constructed from V_{core} in the same way as explained in Section 3.4.5. The ellipsoid for performing the integration is built by using the shape provided by the model. In the case of the weak reflections, the ellipsoid is centered at the position predicted by the orientation matrix instead of computing the centroid from the data. The final integrated intensity of the weak reflection I_w is derived from the intensity enclosed by the core region, as:

$$I_w = \frac{\tilde{I}_{core} - B_4 V_{core}}{x_{core}} \quad (16)$$

where \tilde{I}_{core} is the total number of counts in the core, B_4 is the background obtained from averaging in the outer shell and x_{core} is the fraction of the integrated intensity enclosed by the core. $\sigma(I_w)$ is obtained from Equation (15), setting I/x , I and x to I_w , I_{core} and x_{core} , respectively. In this way, $\sigma(I_w)$ is minimized [22].

If a model is not available, the integration is performed inside an ellipsoid assuming $x = 1$. If there are occupied slots between the first and second neighbors, the ellipsoid of the closest occupied slot is taken as the integration ellipsoid. Otherwise, the default ellipsoid whose dimensions are specified in the input (Figure 4) is taken. The integration proceeds in a similar way as that of the third pass.

4. Results

The basic operation of Int3D will be illustrated following the treatment of the data collected at D19 for the room temperature phase of $[\text{CH}_3\text{NH}_3][\text{Ni}(\text{COOH})_3]$. This compound is a perovskite-like metal-organic compound that at room temperature crystallizes in the orthorhombic $Pnma$ space group, with cell parameters 8.0461(4), 8.2969(4) and 11.5970(8) Å [8]. Throughout this section, we will continuously refer to the series of short videos provided in the supplementary material in order to illustrate the process more clearly.

The first step is to create a project (Video S1). During the creation of the project, we must specify the directory where the scans are stored, and configure the instrument. The instrument parameters refer to the instrument geometry, the detector, the instrument offsets and the wavelength. The instrument geometry and detector parameters usually correspond to those provided by Int3D by default. At this stage, the user only needs to specify the wavelength since the instrument offsets are set later when refining. Once the project is created, a '.pro' file and a '.geom' file are generated and stored in the project directory, together with the four subdirectories mentioned in Section 2.2. The '.pro' file contains the project settings. It is read every time the project is opened. The '.geom' file is the instrument file that stores the instrument configuration and is needed by many of the programs run by Int3D. They can be modified from the **Edit** menu of the GUI.

The data reduction process starts with the peak search (Video S2). The PeakFind interface must be opened from the **Data Reduction** menu. The default values for the threshold and the peak size (Section 3.1), set to 10 and 20 respectively, should work in almost all cases. A pre-visualization of the scans with the 2D viewer (Section 2.1) can be

used to set these values in case problems are found. As every scan in D19 usually collects many reflections, selecting just a few number of scans is usually good enough in order to obtain a good sampling of the reciprocal space. We select the first four scans and we run PeakFind with its default values. During the run several output files are generated in the folder Peaks. The most important one is the '.xyz' file. It contains a line for each reflection found, with the following information: the Cartesian coordinates of the scattering vector, a rough estimate of the intensity, the coordinates of the centroid of the reflection in the direct space (vertical and horizontal pixels and frame), the number of voxels belonging to the reflection and the scan number where the reflection was found. In addition, for each scan PeakFind writes a '.mrk' file. This file is understood by the 2D viewer and can be used to superimposed the centroids found by PeakFind on the raw data. The centroids are shown as crosses, and allow the user to easily check if PeakFind is correctly locating the centroids of the peaks. This is shown in the Video S3. In this video it can also be seen that there are weak reflections that have not been marked. These are reflections for which the threshold and/or peak size constraints were not satisfied.

The next step is to determine the orientation matrix of the crystal (Video S4). The '.xyz' file generated by PeakFind must be loaded from the GetUB interface. When the file is loaded, the reciprocal space viewer displays the set of reflections contained in this file. The user can interact with the set in different ways. The most basic interaction is achieved with the mouse, that allows zooming, translation and 3D-rotation of the set. The reflections can be selected individually or by area. The selection can be hidden, deleted or moved to another set. It is also possible to select reflections in pairs or triplets and measure distances and angles. The search for the orientation matrix takes as input the set of reflections that are visible on the screen. If the threshold and/or peak size values used during the peak search were too low, there could be some spots that do not correspond to real reflections. They can be usually removed by using the slider that filters out reflections by intensity.

The default input parameters (Section 3.2) controlling the search work in most cases, so the user usually only has to run the search without the need for changes. After loading the '.xyz' file, we run the search with all the reflections of the set and the default values provided by Int3D, as shown by Video S4. When the search finishes—usually after a few seconds—GetUB displays the following information for each orientation matrix found: the symmetry, lengths, angles and volume of the unit cell and the R-factor obtained from the indexing of the reflection set with that cell. The cells are arranged in increasing order of the R-factor. The first cell of the table, with parameters 8.0995, 8.2862 and 11.7261 Å and angles very close to 90° corresponds to the searched cell. It can then be selected with the mouse, and display on the reciprocal space viewer by un-checking the box labeled 'hide'. Checking the box labeled as 'Set as reference axes' the reciprocal axes are shown at the bottom-left of the image and the set can be oriented along any arbitrary direction. Video S4 shows that the agreement between the reflection positions and the lattice nodes is rather good. Once a good orientation matrix has been found, it has to be saved and loaded into the crystal object. This will write a '.cry' file in the root directory of the project, and this file will be used for all the programs that need an orientation matrix as input. For instance, the 2D viewer checks if an orientation matrix is available when displaying a scan. If so, then when the mouse moves over the detector the Miller indices of the pixel on which the mouse is positioned are automatically calculated. Pixel indexing is shown in Video S5.

After setting the orientation matrix, the cell parameters and the instrument zero shifts can be refined (Video S6). With this aim, PeakFind is rerun. It is now convenient to select more scans than in the first run to obtain a very good sampling of the reciprocal space. As the calculation is quite fast, we select all the scans of the experiment. Before running, PeakFind checks if an orientation matrix has been set. If so, during the run it also writes a '.raf' file in the Peaks directory, that contains the reflections with their Miller indexes. This file is the required input file for the refinement program RefUB. When PeakFind is finished, RefUB is open from the **Data Reduction** menu. As we know that the sample has orthorhombic symmetry, we fix the cell angles to 90° and do not check their associated

boxes to keep fixed these values during the refinement. The refinement is launched and it converges after a few seconds. The refined parameters together with their old values are displayed on the terminal, and the user is prompted to accept or reject the new values. In case of acceptance, the orientation matrix and the instrument offsets stored, respectively, in the objects Crystal and Instrument are updated. In the Video S6 it can be seen that the refined values are 8.0408(4), 8.2904(4) and 11.5846(8) Å, in very good agreement with the reported values mentioned at the beginning of this section.

The last step is the integration of the intensity of the reflections (Video S7). As in previous tasks, Int3D provides reasonable values for the set of input parameters (Section 3.4), that should work fine for ordinary problems. The integration is the most delicate task in the data reduction process. For this reason, during the run—about 2 min in the example shown in this work—Int3D generates a considerable number of output files that allow the user to easily test the integration. For each integrated scan, Int3D generates the following files: an '.lpt' file that contains the details of each of the integration steps (Sections 3.4.1–3.4.6) for each of the reflections present in the scan; a '.mrk' file, similar to that generated by PeakFind, that allows visualization of the predicted positions of the reflections and test in this way the orientation matrix; a '.box' file for visualizing the integration boxes; a '.msk' file for visualizing the applied masks; a '_2d.elp' file that allows visualization of the sections of the core and total ellipsoids in 2D (Section 3.4.5) and a '_3d.elp' file for visualizing the integration ellipsoids in 3D. Figure 6 is a snapshot where most of these elements are shown for one of the measured scans. For each integration box, its center is marked with a plus symbol. This is the expected position of the center of the reflection on the detector. At the frame where the maximum intensity is expected to be found, the viewer adds a cross symbol, as can be seen for one reflection in the figure. The masks are represented by white pixels. The ellipsoids sections with green and orange borders correspond to the core and total ellipsoids, respectively. It can be seen that there are some ellipsoids that seem not to be well-centered. This is because the frame corresponding to the snapshot do not correspond to the frame of the centroid of the reflection. The ellipsoids, as well as the masks, move along the detector as we move through the scan. This movement is shown in Video S8. Video S9 shows how the integration ellipsoids can be visualized in 3D. Figure 6 also shows some boxes for which no reflection is visible. This is due to the fact that we have integrated the reflections using the space group $P1$, which has a lower symmetry than the correct space group, $Pnma$.

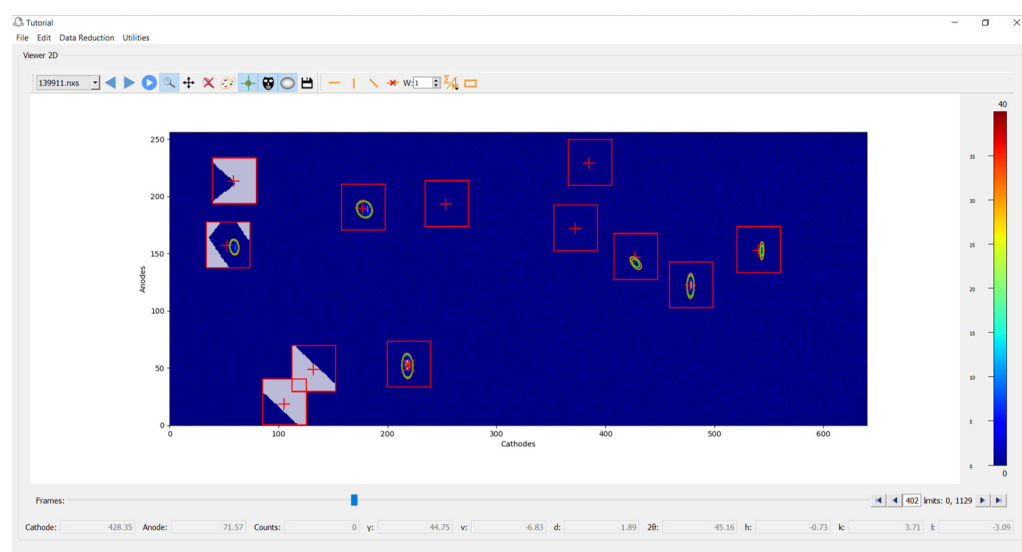


Figure 6. 2D view of a measured scan, showing the integration boxes, masks, ellipsoids and predicted reflection positions. Video S8 shows the dynamic behavior of these elements as we move through the scan.

Int3D also generates ‘.xyz’ and ‘.raf’ files equivalent to those generated by PeakFind, so they can be processed with GetUB and RefUB. As far as strong reflections are concerned, the centroids computed by PeakFind and the integration are almost equivalent. Regarding weak reflections, the integration obviously provides more accurate values.

During the first run of the integration, the library is built (Section 3.4.5). After the integration, the library is written in a ‘.lib’ file, that can be visualized as explained in Section 3.4.5. It could have happened that some of the weak reflections had been integrated when the corresponding library slot was still empty. For this reason, it is always convenient to perform a second integration to ensure that all weak reflections that can be integrated with a model have been integrated in this way. Before this second integration, the ‘.raf’ file generated by the first integration can be used to carry out a second refinement.

The integrated intensities are written in a ‘.hkl’ reflection data file like that used by SHELXL [24]. Every line contains the Miller indices of a given reflection together with its integrated intensity and error. This file can be processed by usual refinement codes as SHELXL, Jana [25] or FullProf [26]. The structure refinement results for the example presented in this section were published in Ref. [8].

5. Summary and Conclusions

This work has presented a first version of Int3D, a data reduction software for single crystal neutron diffraction. The motivation to start building Int3D has been the need to upgrade and unify the software developed in the past at the ILL, and to continue its development. The upgrade has mainly involved rewriting or replacing old Fortran code. All the crystallographic software embedded in Int3D relies now on the CrysFML library. As the last version of this library is ready to handle general space groups, i.e., crystallographic, magnetic and superspace groups, Int3D will allow avoidance of the supercell approximation in the treatment of incommensurate structures. Working directly with the superspace symmetry will be possible.

The unification aims to use the same application for the different single crystal diffractometers of the ILL. The extension of Int3D to deal with small flat detectors like that of D9 has already been done, and it is currently under testing. Once Int3D is ready for D9, its extension to D10 will be straightforward. Due to its highly modular structure, adapting Int3D to a new instrument should not be a particularly complicated task. The integration of X-ray diffraction images of single crystals in flat detectors can also be achieved with relatively small effort for reading other kind of input images and detector geometries.

The major developments concerned the graphical environment. A GUI has been created with a number of embedded graphical tools that make it easy to test the different steps of the data reduction, in many cases by just visual inspection. These tools will be essential for working with complex problems such as incommensurate structures, diffuse scattering or twins. In addition, Int3D provides reasonable default input values for the different tasks of the data reduction. The user-friendly nature of Int3D will allow non-expert users to process their own data when dealing with conventional problems.

Regarding future developments, there are different lines along which progress has to be made. Sometimes, due to the experimental conditions or to the chemical nature of the sample, absorption corrections need to be applied. A task concerning these corrections has to be added to the **Data Reduction** menu. Another item that we will add to this menu is the invocation of a small GUI for handle the new version of the program DataRed, distributed in the FullProf Suite [26]. This new version of the DataRed program is able to handle the raw integrated intensity files generated by Int3D, using Shubnikov and superspace groups to analyze extinction conditions, average equivalent reflections and calculate internal R-factors to assess the quality of obtained data.

In complex problems where diffuse scattering can be important, it will be very helpful to have the possibility to visualize all the collected data in the reciprocal space and to perform slices in this space. A program that combines all the measured scans and generates a single Nexus file with the raw intensity expressed in the reciprocal space has been already

written. It will be interfaced with Int3D and in particular with the reciprocal space viewer of GetUB in the near future.

Finally, the modular structure of Int3D makes it easy to add new integration algorithms. An algorithm based on projecting the three-dimensional scans along the scan direction and integrating the resulting one-dimensional profile by least-squares fitting, is under development.

The code will be public in the near future in the Git repository of the ILL [27], after some further testing and documentation. In the meantime, the code can be obtained by contacting the authors.

Supplementary Materials: The following are available at <https://www.mdpi.com/article/10.3390/cryst11080897/s1>, Videos S1–S9: A series of short videos showing the different functionalities and graphical tools of the application.

Author Contributions: The computing codes have been developed by N.A.K. and J.R.-C.; L.C.-D., O.F. and M.T.F.-D. have participated as testers and suggesting changes. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the FILL2030 initiative, a European Union project within the European Commission's Horizon 2020 Research and Innovation program under grant agreement No. 731096.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to acknowledge the work previously developed at the ILL, that has greatly facilitated the development of some parts of Int3D. Regarding the integration code RETREAT, the authors would like to thank R.F.D. Stansfield, M. Thomas, G. McIntyre, C. Wilkinson, S. Mason and M. Turner. Regarding the different refinement codes, the authors would like to thank M.S. Lehmann, P.J. Brown, A. Barthelemy, and especially A. Filhol.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Newnham, R.E. *Properties of Materials: Anisotropy, Symmetry, Structure*; Oxford University Press: Oxford, UK, 2005.
2. Dove, M.T. *Structure and Dynamics: An Atomic View of Materials*; Oxford University Press: Oxford, UK, 2003.
3. Agilent. *CrysAlis PRO*; Agilent Technologies Ltd.: Yarnton, Oxfordshire, UK, 2014.
4. Bruker. *APEX3*; Bruker AXS Inc.: Madison, WI, USA, 2016.
5. Minor, W.; Cymborowski, M.; Otwinowski, Z.; Chruszcz, M. HKL-3000: The integration of data reduction and structure solution—From diffraction images to an initial model in minutes. *Acta Cryst.* **2006**, *D62*, 859–866. [[CrossRef](#)] [[PubMed](#)]
6. Winter, G.; Waterman, D.G.; Parkhurst, J.M.; Brewster, A.S.; Gildea, R.J.; Gerstel, M.; Fuentes-Montero, L.; Vollmar, M.; Michels-Clark, T.; Young, I.D.; et al. DIALS: Implementation and evaluation of a new integration package. *Acta Cryst.* **2018**, *D74*, 85–97. [[CrossRef](#)] [[PubMed](#)]
7. Langan P.; Greene, G. Protein crystallography with spallation neutrons: Collecting and processing wavelength-resolved Laue protein data. *J. Appl. Cryst.* **2004**, *37*, 253–257. [[CrossRef](#)]
8. Cañadillas-Delgado, L.; Mazzuca, L.; Fabelo, O.; Rodríguez-Carvajal, J.; Petricek, V. Experimental Evidence of the Coexistence of Proper Magnetic and Structural Incommensurability on the $[\text{CH}_3\text{NH}_3][\text{Ni}(\text{COOH})_3]$ Compound. *Inorg. Chem.* **2020**, *59*, 17896–17905. [[CrossRef](#)] [[PubMed](#)]
9. Available online: <https://www.ill.eu/users/instruments/instruments-list/d19/description/instrument-layout> (accessed on 30 July 2021).
10. Available online: <https://www.ill.eu/users/instruments/instruments-list/d9/description/instrument-layout> (accessed on 30 July 2021).
11. Available online: <https://www.ill.eu/users/instruments/instruments-list/d10/description/instrument-layout> (accessed on 30 July 2021).
12. Rodríguez-Velamazán, J.A.; Campo, J.; Rodríguez-Carvajal, J.; Noguera, P. XtremeD—A new neutron diffractometer for high pressures and magnetic fields at ILL developed by Spain. *J. Phys. Conf. Ser.* **2011**, *325*, 012010. [[CrossRef](#)]

13. Available online: <https://www.doi.org/10.5281/zenodo.591709> (accessed on 30 July 2021)
14. Schroeder, W.; Martin, K.; Lorensen, B. *The Visualization Toolkit*, 4th ed.; Kitware: New York, NY, USA, 2006.
15. González-Platas, J.; Katcho, N.A.; Rodríguez-Carvajal, J. CrysFML08: A Modern Fortran Library for Crystallography. **2021**, Unpublished work
16. Available online: <https://github.com/ylikx/forpy> (accessed on 30 July 2021).
17. Maddison, D.R.; Swofford, D.L.; Maddison, W.P. Nexus: An Extensible File Format for Systematic Information. *Syst. Biol.* **1997**, *46*, 590–621. [[CrossRef](#)] [[PubMed](#)]
18. Busing, W.R.; Levy H.A. Angle Calculations for 3- and 4- Circle X-ray and Neutron Diffractometers. *Acta Cryst.* **1967**, *22*, 457–464. [[CrossRef](#)]
19. Duisenberg, A.J.M. Indexing in Single-Crystal Diffractometry with an Obstinate List of Reflections. *J. Appl. Cryst.* **1992**, *25*, 92–96. [[CrossRef](#)]
20. Krivy, I.; Gruber, B. A Unified Algorithm for Determining the Reduced (Niggli) Cell. *Acta Cryst.* **1976**, *A32*, 297–298. [[CrossRef](#)]
21. Le Page, Y. The Derivation of the Axes of the Conventional Unit Cell from the Dimensions of the Buerger-Reduced Cell. *J. Appl. Cryst.* **1982**, *15*, 255–259. [[CrossRef](#)]
22. Wilkinson, C.; Khamis, H.W.; Stansfield, R.F.D.; McIntyre, G.J. Integration of Single-Crystal Reflections Using Area Multidetectors. *J. Appl. Cryst.* **1988**, *21*, 471–478. [[CrossRef](#)]
23. Lehmann, M.S.; Larsen, F.K. A Method for Location of the Peaks in Step-Scan-Measured Bragg Reflexions. *Acta Cryst.* **1974**, *A30*, 580–584. [[CrossRef](#)]
24. Sheldrick, G.M. Crystal structure refinement with SHELXL. *Acta Crystallogr. Sect. C Struct. Chem* **2015**, *C71*, 3–8. [[CrossRef](#)] [[PubMed](#)]
25. Petricek, V.; Dusek, M.; Palatinus, L. Crystallographic Computing System JANA2006: General features. *Z. Kristallogr.-Cryst. Matter* **2014**, *229*, 345–352. [[CrossRef](#)]
26. Rodríguez-Carvajal, J. Recent advances in magnetic structure determination by neutron powder diffraction. *Phys. B* **1993**, *192*, 55–69. Available online: <https://www.ill.eu/sites/fullprof/php/downloads.html> (accessed on 30 July 2021). [[CrossRef](#)]
27. Available online: <https://code.ill.fr/explore/projects> (accessed on 30 July 2021).