Manual

singlepowder

Integrating single-crystal area-detector data as powder diffractogram

Tobias Fröhlich

Contents

Installation (Linux)

Requirements

- C++11
- std library
- cmake

- Google-Test

Google-Test is installed as follows:

```
sudo aptitude install libgtest-dev
cd /usr/src/googletest/
sudo cmake . \\
sudo cmake --build . --target install
```

Build

In the directory `singlepowder/build/`:

```
cmake ..
make
```

Testing

In the directory `singlepowder/bin/`:

```
./Test
```

Installation

In the directory `singlepowder/bin/`:

```
sudo cp singlepowder /usr/bin/
```

Usage

Integration

Running

For integrating, singlepowder is run with the command `integrate` and one further parameter, that is the name of the parameter file, i.e. parameters.txt:

```
singlepowder integrate parameters.txt
```

Input files

Figure 1 shows an example for input files. The directory
`~/singlepowder_test/TD015S001apex004/`
contains the data files `TD015S001apex004_01_0001.out`, `TD015S001apex004_02_0001.out`, etc.
These files are listed in `~/singlepowder_test/list.txt`. This list file contains one line per image file. So far, only the columns filename, detectordistance and weight are used.
The program only needs the name of the parameter file `parameters.txt` and gets all further information from this.
The output is written to the file given in the line `output_filename` of the parameter file. In the example, the output file is named `output.txt`.
The file name of the mask file is given by the parameter `mask_filename`.
Everything behind the character `#` in the parameter file or the list file is a comment and ignored by the program. Empty lines or lines consisting of only a comment are allowed. The order of the parameters in the parameter file does not matter. Appart from comments, every line in the parameter file must consist of exactly two words. Thus, spaces in file names are not allowed.
The parameters in the parameter file are the following (all parameters must be given, there are no default values):

`pixel_width`	Width of one pixel in mm
`pixel_height`	Height of one pixel in mm
`centre_pixel_x`	x index of the pixel hit by the direct
`centre_pixel_y`	y beam at $2\theta = 0$ (non-integer index possible)
`angle_min`	The output powder diffractogram
`angle_max`	covers 2θ from `angle_min` to `angle_max`
`step`	with stepsize `step`.
`image_list_filename`	path and name of the list file
`data_directory`	path for the data files
`output_filename`	path and name of the output file
`output_format`	format of the output file (standard or detailed)

Creating a mask

The mask is created with the following steps:

- Averaging many detector images (preferrably without any reflections but just background).
- Inverting the values.

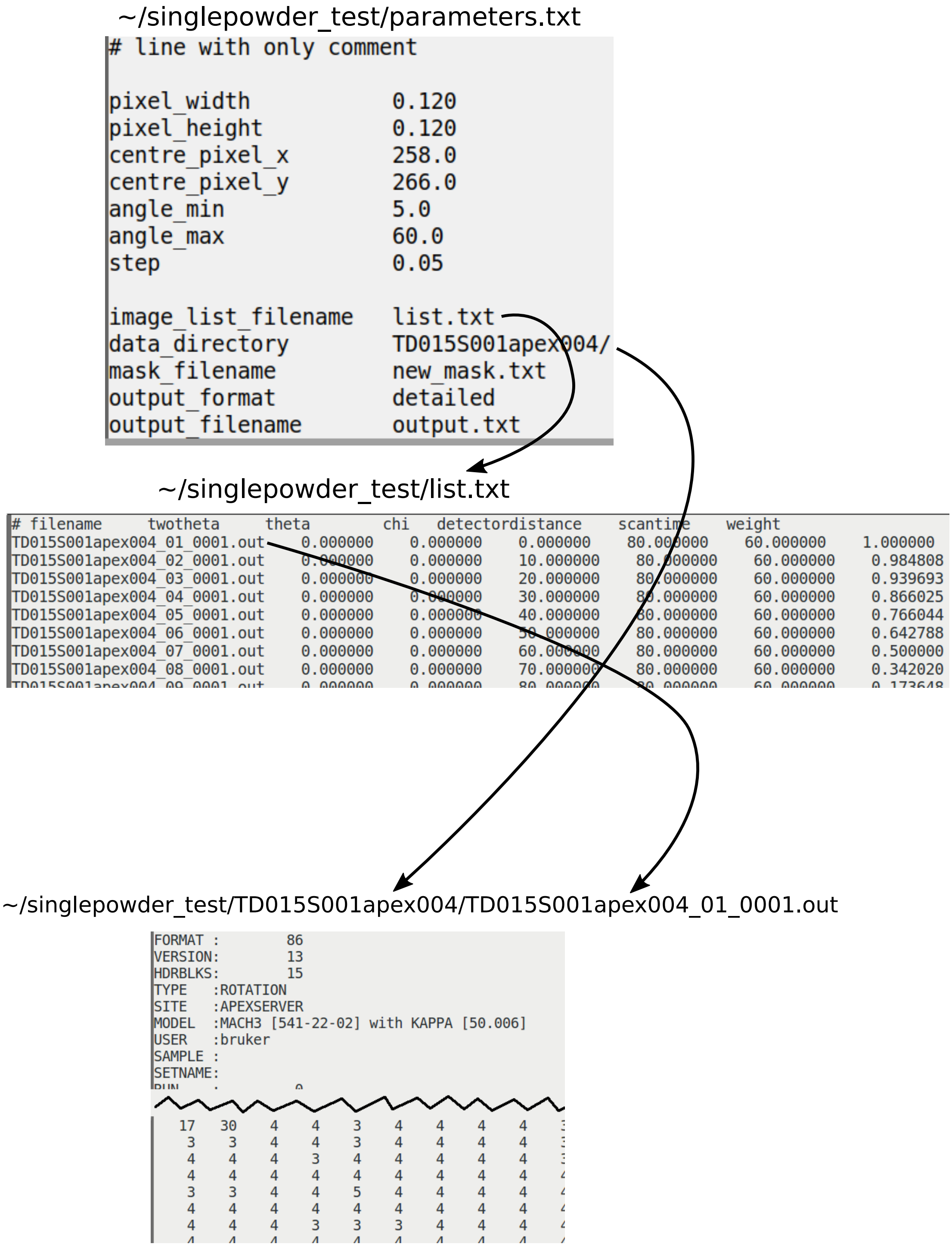

Figure 1: Example for input files. In the directory `~/singlepowder_test/`, the command `singlepowder parameters.txt` runs the program.

- Setting the value for corrupted pixels to zero.

Averaging is done using the command

```
singlepowder make_mask <data_directory> <list_filename> <output_filename>
```

Iverting uses the command

```
singlepowder invert_mask <old_mask_filename> <new_mask_filename>
```

For setting currupted pixels to zero, multiply with a mask with 0 for the corrupted pixels and 1 for the others:

```
singlepowder multiply_masks <mask_filename1> <mask_filename2> <output_mask_filename>
```

Example: Assuming, the file `corrupted.txt` contains a mask with corrupted pixels set to 0 and others to 1, the directory `TD015S001apex004` contains the data files and `list.txt` is the list file, the following commands must be executed:

```
singlepowder make_mask TD015S001apex004/ list.txt averaged.txt
singlepowder invert_mask averaged.txt inverted.txt
singlepowder multiply_mask corrupted.txt inverted.txt mask.txt
```

Details

Geometry of the diffractometer

All lengths are in mm and all angles in deg.

Figure 2 shows a four circle diffractometer. The angles are shown with the directions used in the program (only 2θ is actually used so far). In order to avoid confusion, 2θ names the position of the detector while "powder angle" ε is used for the angle between the diffracted beam and the direct beam for a certain pixel. The pixel indices x and y and the pixel width w and height h are shown with the direction used by the classes Geometry and DetectorImage.

The variables in the figure and the following calculation refer to the following variables in the program code:

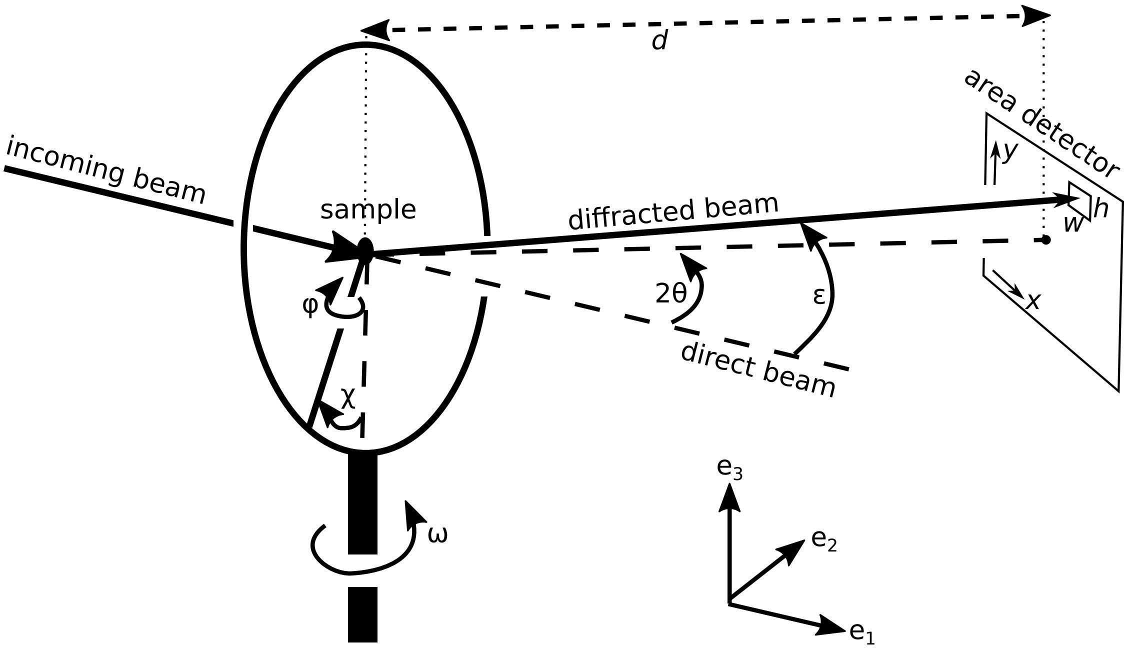

Figure 2: Direction of the angles 2θ, ω, χ, φ, the pixel indices x, y and the Cartesian basis vectors e_1, e_2, e_3.

d :	detector_distance	(mm)
2θ :	twotheta	(deg)
ε :	powder_angle	(deg)
x :	pixel_x	(index)
y :	pixel_y	(index)
w :	pixel_width	(mm)
h :	pixel_height	(mm)
x_c :	centre_pixel_x	(index)
y_c :	centre_pixel_y	(index)
Δx :	delta_x	(mm)
Δy :	delta_y	(mm)

x_c and y_c are the indices of the pixel that is hit by the direct beam when all angles are set to zero. The deviation (in mm) from this pixel is for the pixel with indices (x, y):

$$\Delta x = w(x - x_c)$$
$$\Delta y = h(y - y_c)$$

Using the basis vectors[1] defined in Figure 2 (the origin is placed at the pivot point of the goniometer, i.e. the sample), we get for $2\theta = 0$ the following coordinates of the pixel:

$$\boldsymbol{p} = \begin{pmatrix} d \\ -\Delta x \\ \Delta y \end{pmatrix}$$

The rotation matrix around e_3 depends on 2θ:

$$R = \begin{pmatrix} \cos(2\theta) & -\sin(2\theta) & 0 \\ \sin(2\theta) & \cos(2\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The coordinates of the pixel with indices (x, y) depend on d and 2θ and are thus:

$$\begin{aligned} \boldsymbol{p'} &= R\boldsymbol{p} \\ &= \begin{pmatrix} d\cos(2\theta) + \Delta x \sin(2\theta) \\ d\sin(2\theta) - \Delta x \cos(2\theta) \\ \Delta y \end{pmatrix} \end{aligned}$$

The direct beam intersects the detector circle in the following point:

$$\boldsymbol{r} = \begin{pmatrix} d \\ 0 \\ 0 \end{pmatrix}$$

For the angle ε between the direct and diffracted beam, the following condition holds:

$$|\boldsymbol{r}||\boldsymbol{p}| \cos(\varepsilon) = \boldsymbol{r} \cdot \boldsymbol{p'} \quad ,$$

where $\cdot$ denotes the scalar product.
From this, we can calculate the powder_angle ε :

[1]The basis vectors are dimensionless and the coordinates have unit mm.

$$
\begin{aligned}
\varepsilon &= \arccos \frac{\boldsymbol{r} \cdot \boldsymbol{p}'}{|\boldsymbol{r}||\boldsymbol{p}'|} \\
&= \arccos \frac{d(d\cos(2\theta) + \Delta x \sin(2\theta))}{\sqrt{(d\cos(2\theta) + \Delta x \sin(2\theta))^2 + (d\sin(2\theta) - \Delta x \cos(2\theta))^2}\ d} \\
&= \arccos \frac{d\cos(2\theta) + \Delta x \sin(2\theta)}{\sqrt{(d\cos(2\theta) + \Delta x \sin(2\theta))^2 + (d\sin(2\theta) - \Delta x \cos(2\theta))^2}}
\end{aligned}
$$

This is the formula in Geometry::calculate_powderangle().

Integration and error propagation

The algorithm loops over all pixels of all detector images. For each detector image, the detector distance and 2θ are given in the list file. From the geometric parameters, the powder_angle is calculated. The counts of this pixel are summed in the diffractogram at the according powder_angle. The bin of the histogram is used, where the angle deviates maximally `step`/2. If the powder_angle of the pixel lies outside the interval $[\texttt{angle\_min} - \texttt{step}/2, \texttt{angle\_max} + \texttt{step}/2]$, the counts are discarded. The counts are weighted by the weight given in the list file.
Let i, j run over all pixels of all detector images for a fixed powder_angle. The counts are c_i and the weights w_i. A weight w_i is the product of the weight of the image (given in the list file) and the weight of the pixel (given in the mask file). The intensity at this angle is:

$$
I = \frac{\sum\limits_{i=1}^{n} w_i c_i}{\sum\limits_{j=1}^{n} w_j}
$$

The error on the counts are $\sigma_{c_i} = \sqrt{c_i}$, so the error on the intensity can be computed in the following way:

$$
\begin{aligned}
\sigma_I &= \sqrt{\sum_{i=1}^{n} \left(\frac{\partial I}{\partial c_i}\right)^2 \sigma_{c_i}^2} \\
&= \sqrt{\sum_{i=1}^{n} \left(w_i \middle/ \sum_{j=1}^{n} w_j\right)^2 c_i} \\
&= \frac{1}{\sum\limits_{j=1}^{n} w_j} \sqrt{\sum_{i=1}^{n} w_i^2 c_i}
\end{aligned}
$$

During the integration, the following sums are collected for each powder_angle:

$$\texttt{sum\_of\_weights} = \sum_{i=1}^{n} w_i$$
$$\texttt{sum\_of\_weighted\_counts} = \sum_{i=1}^{n} w_i c_i$$
$$\texttt{sum\_of\_squareweighted\_counts} = \sum_{i=1}^{n} w_i^2 c_i$$

These values are written to the output file when the parameter `output_format` in the parameter file is set to detailed.
The sums are calculated in Diffractogram::add_counts().
The intensity and its error is then calculated as follows:

$$\texttt{intensity} = \quad I = \quad \texttt{sum\_of\_weighted\_counts} \Big/ \texttt{sum\_of\_weights}$$
$$\texttt{error} = \quad \sigma_I = \quad \sqrt{\texttt{sum\_of\_squareweighted\_counts}} \Big/ \texttt{sum\_of\_weights}$$

This is the formula in Diffractogram::calculate_intensities_and_errors().

Structure of the program

The structure of the program for integration is shown in Figure 3. For the creation of the mask, the class MaskMaker is used.

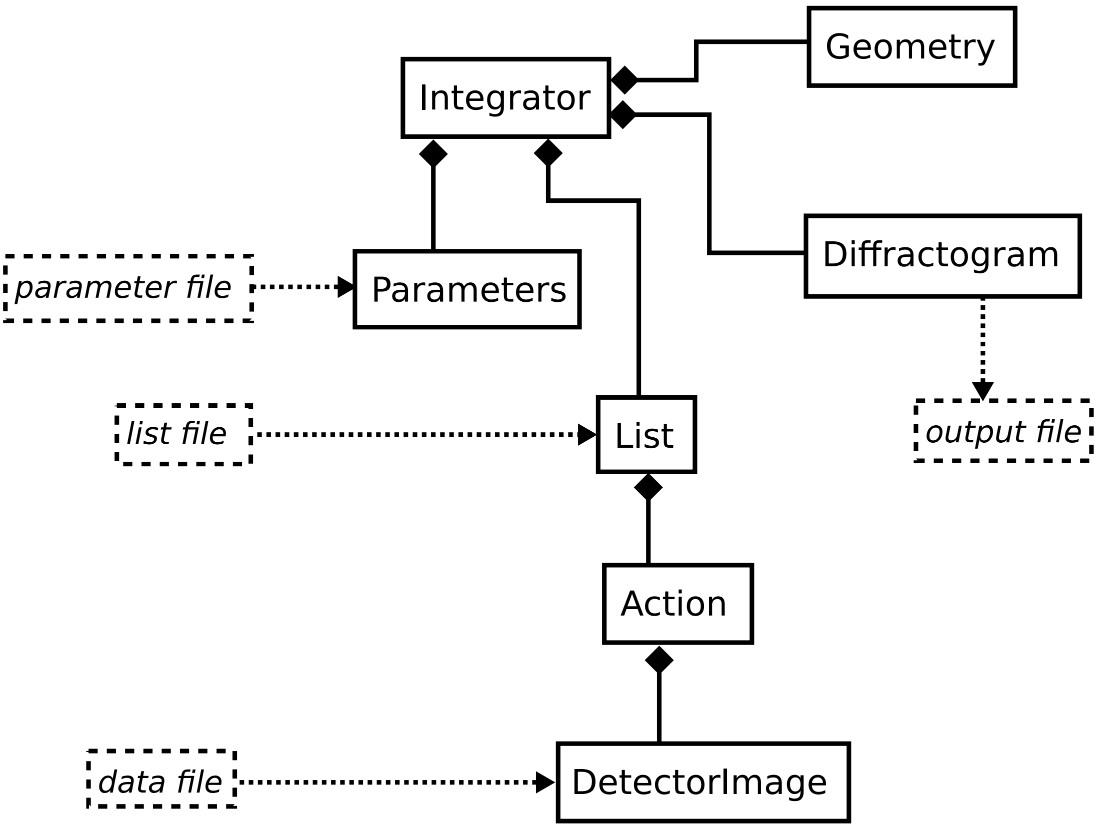

Figure 3: Diagram showing the hierarchy of the classes and the files they read and write.