Internship report

GED ready - a graphical user Interface for GED data reduction

Till Westermann

Contents

Acknowledgment

I would like to thank Dr. Raphael Berger for giving me the opportunity to work on this interesting topic, for the introduction to the topic and all the helpful discussions. I also would like to thank Dr. Stuard for the discussions
All staff of the chair of inorganic chemistry are acknowledged for the very good working environment and supporting this work.
The chair of theoretical chemistry is acknowledged for providing the computer resources for this work.

1 Introduction

Gas phase Electron Defraction (GED) is a state of the art method for determining molecular structures in gas phase. Even 60 (CHECK!) years after its invention, analysis of the experimental data is not trivial or automated as it is in X-Ray defraction. An example for these experimental data is visualisded in Figure **??** (p. **??**). To determine the limits or angle for the data reduction human intuition is still required. The aim of the written program is to provide an convenient way to graphically determine all parameters (limits, angle etc.) for the data reduction. For convenience, the PIMAC (REF!) program was implemented into the graphical program for direct data reduction. Also some minor changes on the PIMAC program were made for compatibility reasons.
This documentation will first explain the graphical user interface of *GED ready* and the workflow (Chapter **??** and **??**). In Chapter **??** the technical details of the program itself will be discussed in detail.

2 Description of the user interface

The graphical user interface is very easy to use. An example is given in Figure ?? (p. ??) how the interface looks like.

2.1 Input

The program can deal with all kind of image formats. It was tested with .tif files but should work with all other formats, as long as they are supported by Qt.
The the images can be opened with the OpenFile menu.
The following should be considered:

1. The program assumes that the data files (.img) are in the same directory then the image files (.tif).

2. The pimag executable must be in the same directory then the .tif and .img files. Otherwise the data reduction (see later) will not work.

2.2 Workflow

A complete workflow is described in Figure ?? (p. ??). The workflow describes one way to work with the program. Theoretically it is possible to go back at any point of the diagram.

2.3 Output

The program hast tree modes to write out input files for pimag. For all images imported, for all images selected in the list or for all images flagged as usable.
For every image one .txt file is written out to the same directory then the image files.
In the end a bash-script is written do reduce.sh that can be used for data reduction. Notice that it must be executable (chmod +x) to execute it. Assuming the .tif and .img files are in /home/geduser/mydata/, the following has to be done:

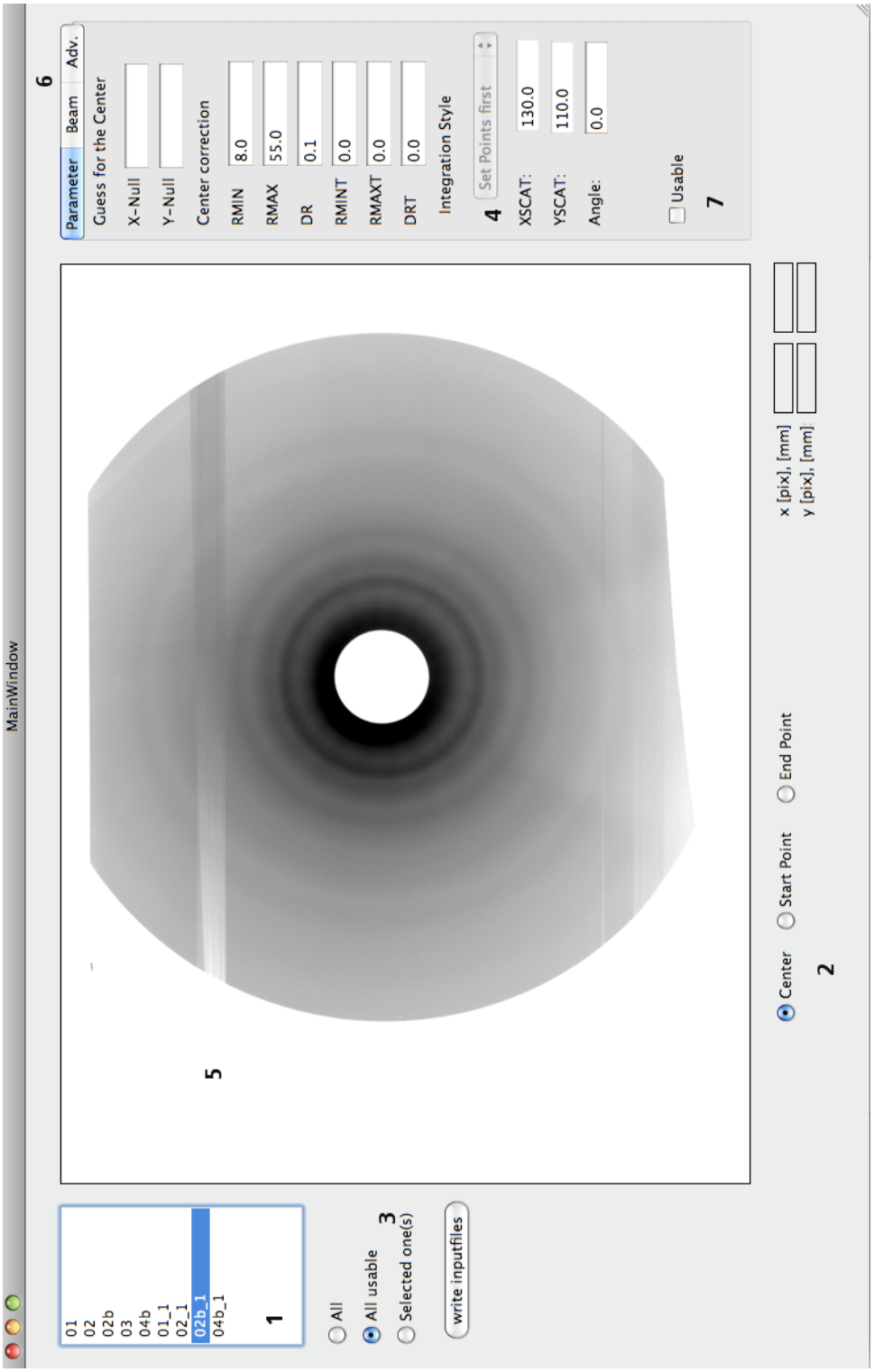

Fig. 1: Typical screenshot of the user interface. (1) The itemlist (listWidget), (2) Radiobuttons for selecting points, (3) Radiobuttons for selecting which input files to write out, (4) drop-down menu for selecting data reduction style, (5) main graphic, (6) tabs to set more parameters and (7) the usable checkbox.

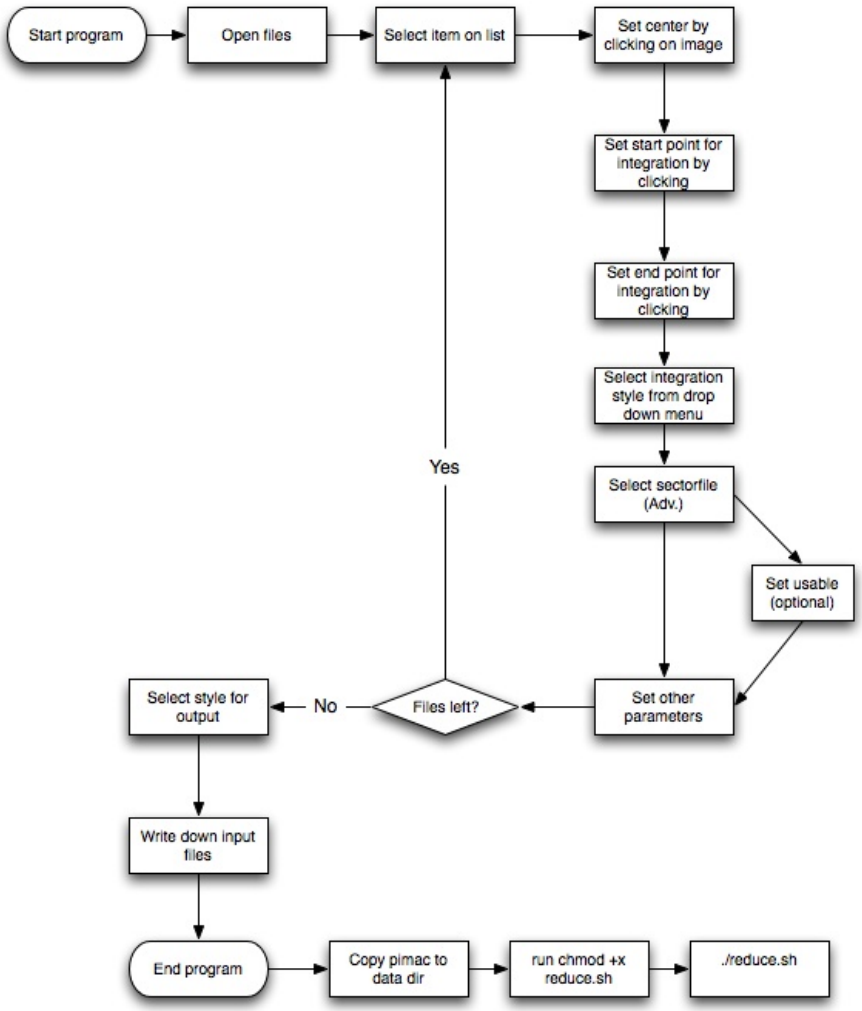

Fig. 2: Workflow of the program.The "set usable"-step is optional, depending on the way the input files are written out later. If you decide to write out input files for every image, it is not necessary to set.

```
cd /home/geduser/mydata/
chmod +x reduce.sh
./reduce.sh
```

The last command starts the data reduction. The script creates (or overrides!) the following files, assuming that $filename was the name of the .tif file, without the .tif.

$filename.curv This file is used for further processing of the data.

$filename.plot This file is used for plotting, a free program like xmgr [**?**] or GNUPLOT [**?**] can be used.

3 Technical documentation

3.1 Program structure

The program is written in C++ using Qt 4.5 (documentation online: [**?**]).
Data management is done via a QListWidget using GEDItem as a custom class for the Items of the list. GEDItem is inherited from QListWidgetItem. Most explanations are written as comments in the source code, nevertheless the most important custom memberfunctions and properties will be discussed in detail.

3.2 The GEDItem class

The definition of the GEDItem class read as:

```
1 class GEDItem : public QListWidgetItem
```

All inherited functions, slots, signals and properties are documented online [?].

3.2.1 Properties

The (private) properties of the GEDItem class are:

```
1 private:
2         QString IPLA, IXMA, JYMA;
3         QString PIXEL, XPIXFA, YPIXFA, XSCAT, YSCAT, XNULL,
              YNULL, RMIN, RMAX, DR, RMINT, RMAXT, DRT, TUNEXP,
              RADI, CADIST;
4         QString WAVE, DELTAS, SEPLA, ISECT, IRECOA, IRECOA2,
              ANGLE, SECFI;
5         bool Useable;
6         QString Path, Mode;
7         QString xRMAXT, yRMAXT, xAngle, yAngle;
```

There are get- and setmethods for every private property of the class named getProperty and setProperty.

3.2.2 float GEDItem::distance(float x1, float x2, float y1, float y2)

Gets two points p_1 and p_2 in $\mathbb{R}^2$ and returns the distance between the points.

3.2.3 float GEDItem::calcAngle(float x1, float y1, float x2, float y2, float x3, float y3)

Gets a triangle defined by points p_1, p_2 and p_3 in $\mathbb{R}^2$ and returns the angle in the defined triangle at p_1.

3.2.4 QString GEDItem::writeInputFile()

Writes out an input file for PIMAG [**?**] and returns a QString with the complete filename (including path) of the written file.
The complete filename of the input file is the same then the selected .tif file, but with .txt instead of .tif. The returned filename is used for a list in QMainWindow.

3.3 The QMainWindow Class

This class handles the main window and all of it's slots and signals with the instance ui. The definition reads as:

```
1 class MainWindowClass : public QMainWindow
```

The only notable property of QMainWindow is QStringList fileList that contains the list of all inputfiles that have been written to hard drive.

3.3.1 bool MainWindowClass::eventFilter(QObject * watched, QEvent * event)[slot]

Is the primary eventfilter of the program. It handles all events coming from ui −>picLabel (the picture-frame of the program).
Returns if the events was handled, always false (for internal reason).

3.3.2 bool MainWindowClass::isIn(QString name) [slot]

Returns true, if the name is already in the list of the listWidget.

3.3.3 void MainWindowClass::on_actionOpenFile_triggered() [slot]

Executes if the user clicks on openfile. Handles all renaming of doubled items in the listWidget and ads the selected files to the listWidget as GEDItems.

3.3.4 void MainWindowClass::setValuesByMethod(QString method)

When this function is called, it calculates the values for XSCAT, YSCAT and ANGLE and writes them to the corresponding lineEdits at the user interface.

3.3.5 void MainWindowClass::on_listWidget_ itemClicked (QListWidgetItem * item) [slot]

Is called when the user clicks on an item at the listWidget. It sets all stored values to the lineEdit forms, sets the picture and calls setValuesByMethod().

4 References