Internship report

xpimag - a graphical user Interface for GED data reduction

Till Westermann

Contents

1 Introduction

Gas phase Electron Diffraction (GED) is one of only two existing methods for the direct determination of molecular structure parameters in gas phase. Even though the method was introduced in the 1930th, analysis of the experimental data is not trivial or automated as it is in X-Ray diffraction.

GED is based on the diffraction of electrons by randomly orientated molecules in the gas phase. The compound is evaporated into high vacuum ($\approx 10^{-7}$ Torr) though a nozzle where an electron beam crosses the molecular jet.

The electrons are emitted thermally and accelerated by an electric potential. The beam is focused by magnetic or electric fields and then guided into the diffraction chamber. The electron beam is well defined in terms of the wavelengths λ with a fluctuation not bigger than 0.1 %.

Upon interaction of the electron beam with the molecular jet in the diffraction chamber several processes may occurs. Because of the low pressure the probability that an electron is not scattered is 99 %. This directly leads to the reasonable assumption that multiple scattering of one electron can be neglected.

For the rest of the electrons scattering occurs. Neglecting rotational-vibrational coupling, the elastic electron scattering intensity for a molecule with N atoms reads as:

$$I(s) = \frac{K^2 I_0}{R^2} \left(\underbrace{\sum_{i}^{N} |f_i(s)|^2}_{\text{Atomic scattering}} + \underbrace{\sum_{\substack{i,j \\ i \neq j}}^{N} f_i(s) f_i^*(s) \frac{\sin s r_{ij}}{s r_{ij}}}_{\text{Molecular scattering}} \right)$$

Where $s = 2k_0 \sin\theta/2$ is the absolute value of the momentum transfer, θ is the scattering angle, K is a constant, R is the distance of the observation point from the scattering center, $f(s)$ are the electron scattering amplitudes and $r_{ij} = |\vec{r_i} - \vec{r_j}|$ (with $\vec{r_i}$ as the position vector of the atom i in the molecular coordinate system). The atomic scattering is the dominant factor, but its intensity is not relevant to the molecular structure problem since it does not contain any information about the inter-atomic distances.

This model can be expanded to include vibration. In this approximation the

molecular scattering intensity reads as:

$$I_m(s) = \frac{K^2 I_0}{R^2} \sum_{\substack{i,j \\ i \neq j}}^{N} g_{ij}(s) \exp\left(-\frac{1}{2} l_m^2 s^2\right) \frac{\sin(s(r_a - \kappa s^2))}{s r_a}$$

Where $g_{ij}(s) = |f_i(s)||f_j(s)| \cos(\eta_i(s) - \eta_j(s))$, r_a is an effective average internuclear distance, l_m an effective mean vibrational amplitude and κ a asymmetry constant, related to the morse potential.

Photographic or electron image plates are used to record the intensity of the scattered electrons over a given time. In the Bielefeld Group Eu^{2+} doped BaFBr plates are used. The plates record the electron intensity by accumulation an exited state of Eu^{2+}. The intensity is read out with a red laser, which stimulates the emission of a photon. The intensity of the emitted light is detected by a photo-multiplier. An example of the experimental data is shown in Figure 1 (p. 5).

The experimental intensities contained in the form of a 2D distribution, has to be reduced by radial integration and corrected for various experimental factors, and is finally related to the distribution of weighted inter atomic distances by a Fourier transformation. [1]

To reduce the experimental data the PIMAG program was written by T.G. Stand *et al.* [2]. Although PIMAG technically works fine, it is not very user friendly. PIMAG requires an input files with non intuitive parameters. To determine the limits or angles for the data reduction human intuition is still required. The aim of the written program is to provide a convenient way to graphically determine all parameters (limits, angles etc.) for the data reduction and write out input files for PIMAG. For convenience, a bash-script is written out, that calls the data reduction with PIMAG. Also, some minor changes to the PIMAG program were made for compatibility reasons.

The program runs on all Unix, Linux and Mac OS X systems(Windows support should be easy to implement).

This documentation will first explain the graphical user interface of *xpimag* and the work flow (Chapter 2. In Chapter 3 the technical details of the program itself will be discussed in detail.

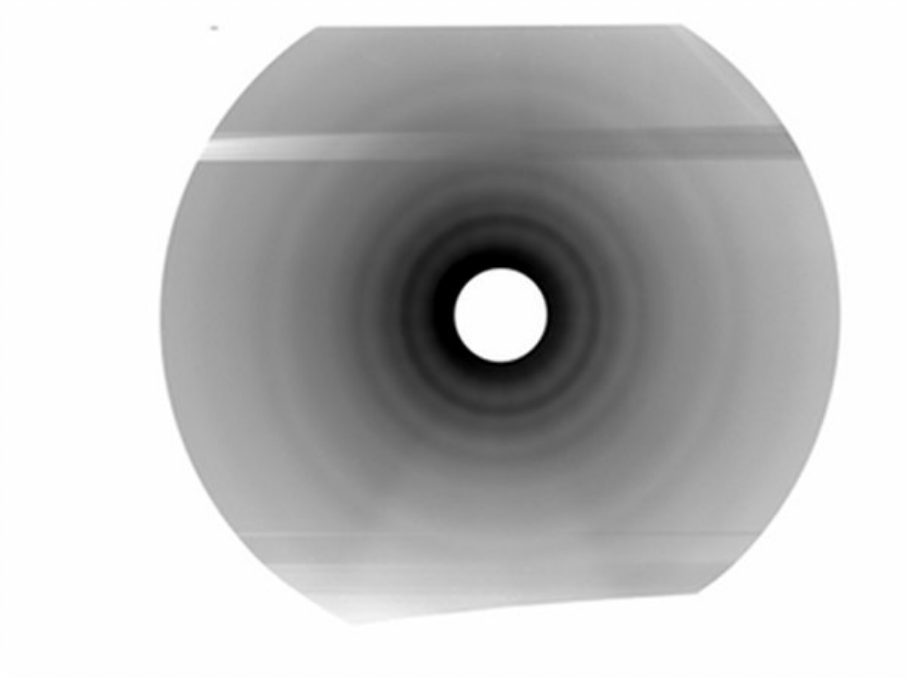

Fig. 1: An example of the raw data from the GED experiment, where an error ocurred during scanning, as can be seen from the irregular horizontal stripe. With the correct parameters, it may still be possible to extract useful information from the image.

2 Description of the user interface

The graphical user interface is very easy to use. An example is given in Figure 2 (p. 7) of how the interface appears.

2.1 Input

The program can deal with all kinds 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 as the image files (.tif).

2. The executable must be in $PATH. Otherwise the data reduction (see later) will not work.

2.2 Work-flow

A complete work-flow is described in Figure 3 (p. 8). The work-flow describes one way to work with the program. Theoretically it is possible to go back at any point of the diagram.

The first click on the graphic sets the approximetly center, the second click sets the start point and the third click sets the endpoint for the integration and an angle. This is also indicated by the radio buttons (See Figure 2 (p. 7). The angle is calculated depending on the selected integration mode.

2.2.1 Integration mode selection

There are several integration modes available:

Whole area This means that a circle with the center and radius determined above will be used for integration. If the measurement was good and there are no shadows on the plate, this is the best choice.

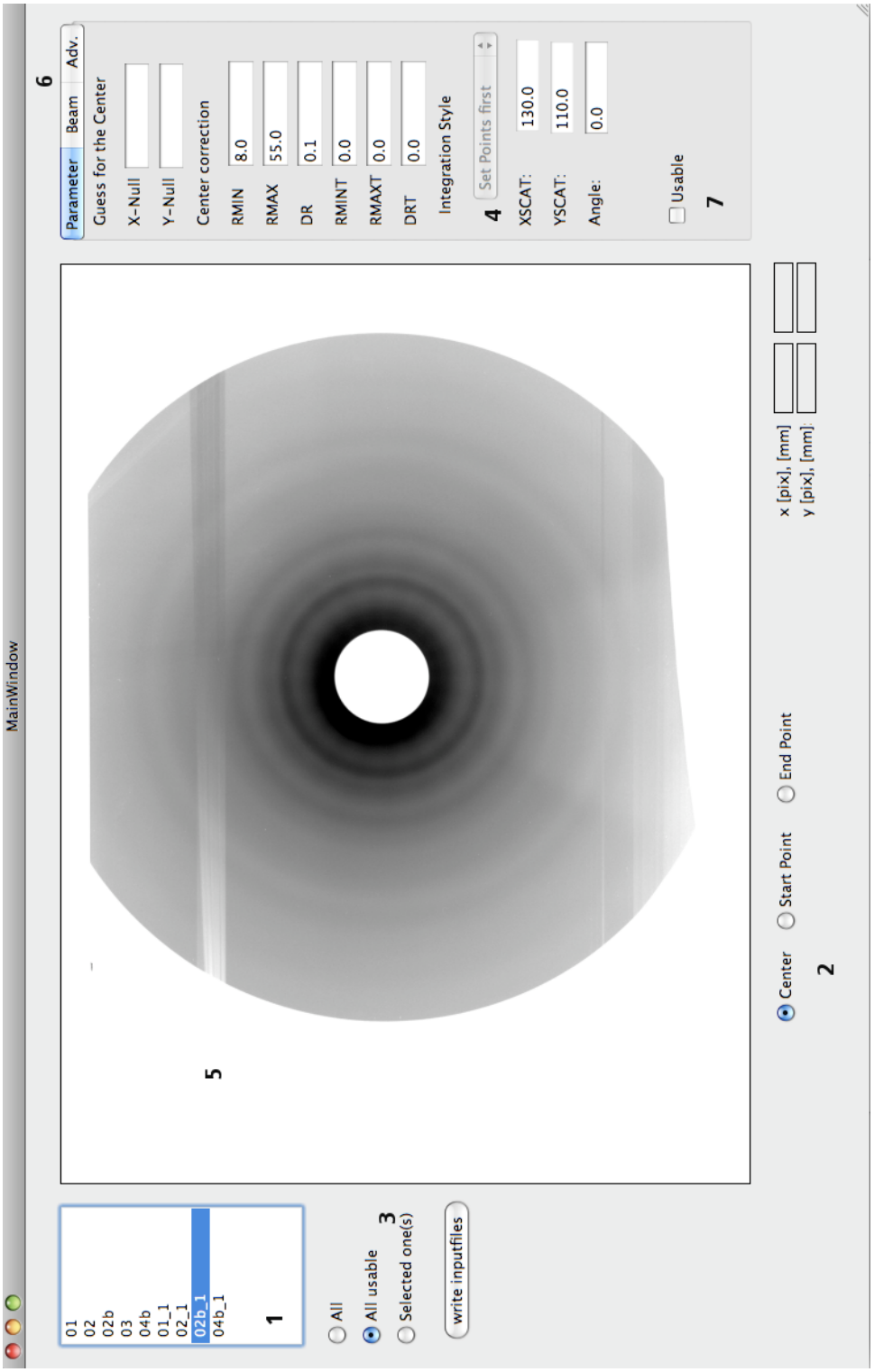

Fig. 2: Typical screen shot of the user interface. (1) The item list (listWidget), (2) Radiobuttons for selecting points, (3) Radiobuttons for selecting which input files to write out, (4) drop-down menu for selecting data reduction style, (5) main graphic, (6) tabs to set more parameters and (7) the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

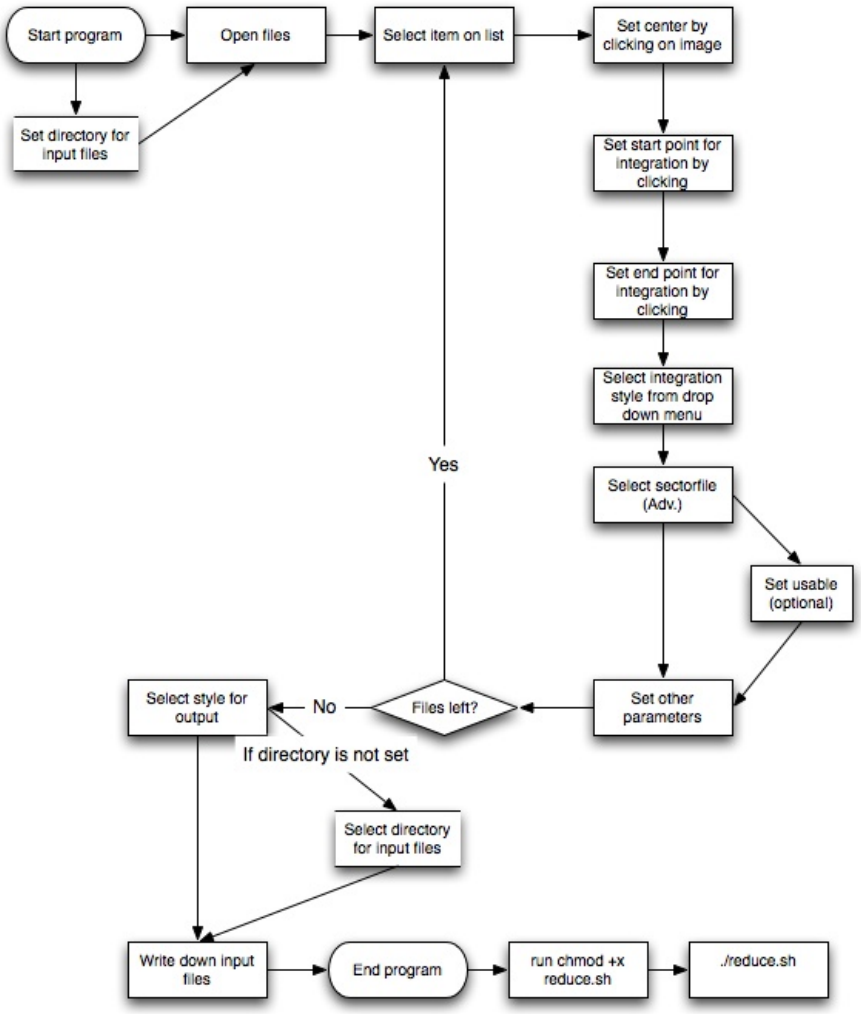

Fig. 3: Work-flow of the program. The "set usable"-step is optional, depending on the way the input files are written out later. This is only necessary if the mode "only usable" is used.

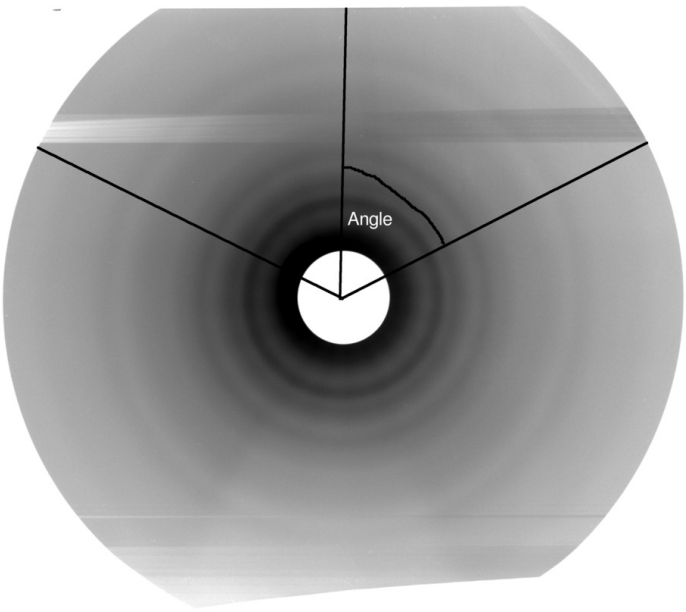

Fig. 4: Used sectors for mode *Top*.

Top The 'Top' half is taken for integration. The angle is determined by the endpoint relative to the y-axis (the center sets the zero point of the y axis). An example is given in Figure 4 (p. 9).

Down The bottom half is taken for integration. The angle is determined as above, see example is given in Figure 5 (p. 10).

Top-Down The 'Top' and 'Down' sectors are taken for integration. The angle is determined in the same way as above. An example is given in Figure 6 (p. 10).

Right The right-hand side is taken for integration. The angle is the endpoint relative to the x-axis (the center is zero). An example is given in Figure 7 (p. 11).

Left Same as Right, but for the left side. See Figure 8 (p. 11).

Right-Left The right- and left-hand sides are both taken into account. See Figure 9 (p. 12).

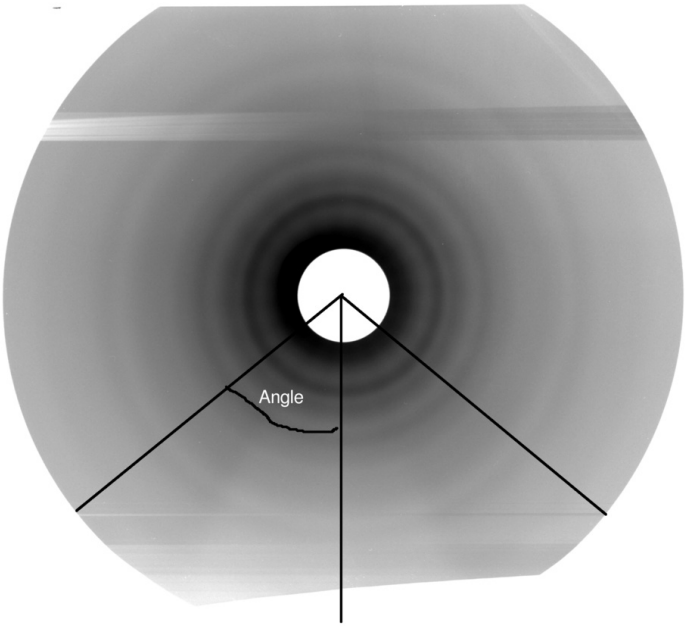

Fig. 5: Used sectors for mode *Down*.

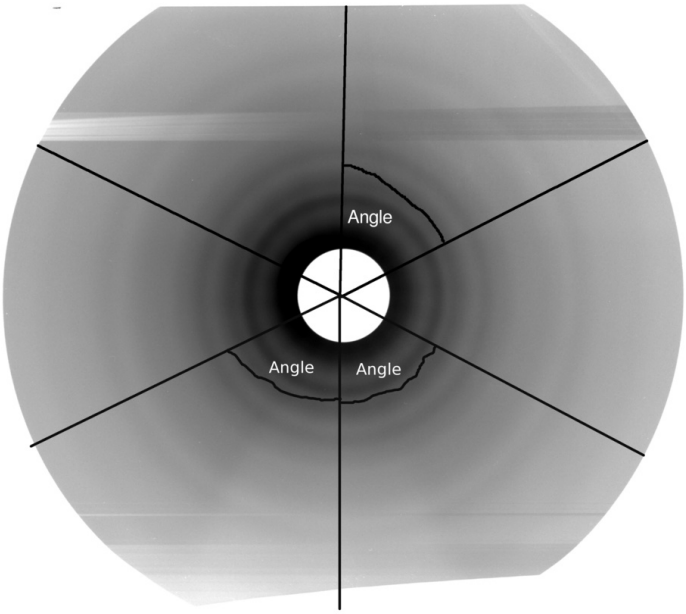

Fig. 6: Used sectors for mode *Top-Down*.

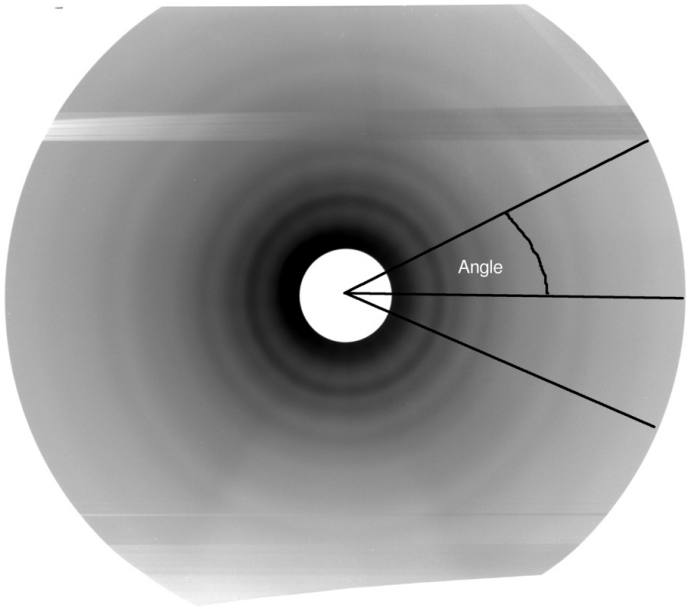

Fig. 7: Used sectors for mode *Right*.

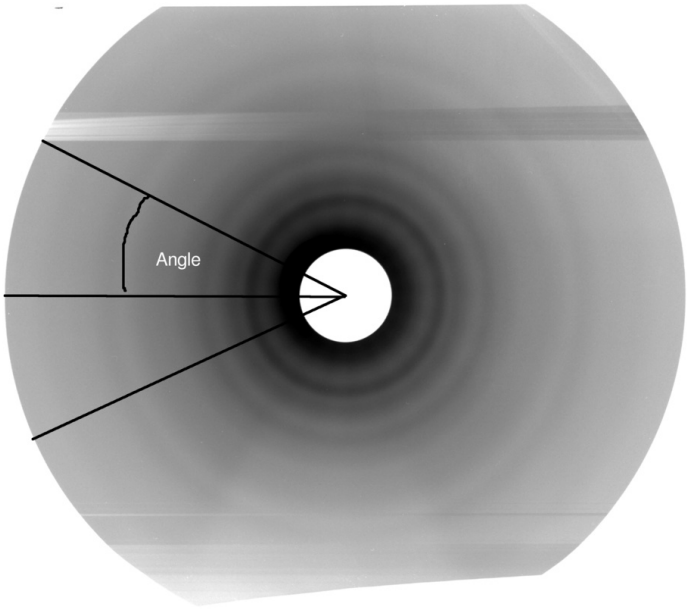

Fig. 8: Used sectors for mode *Left*.

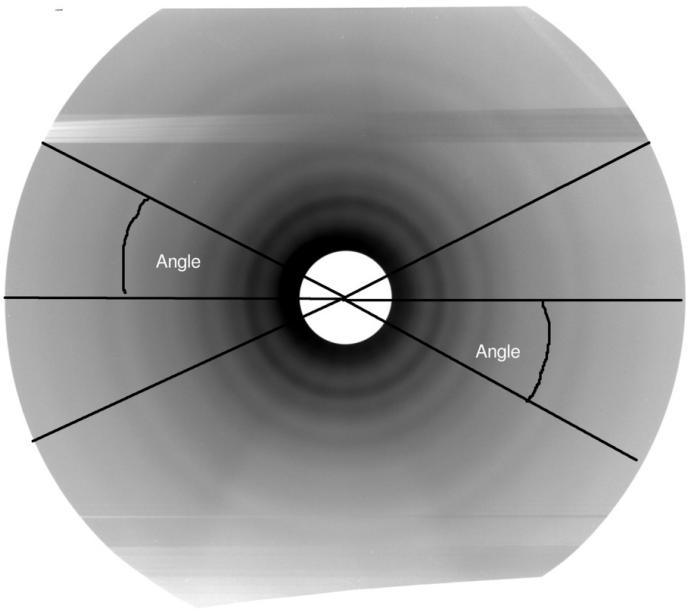

Fig. 9: Used sectors for mode *Right-Left*.

2.3 Output

The program has three 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 as the image files. In the end a bash-script is written to 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:

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

The last command starts the data reduction. The script creates (or overwrites!) 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.dat	This file is used for plotting, and can be read by a free program like xmgr [3] or GNUPLOT [4].
$filename.info	This file contains detailed information about the data reduction process.

3 Technical documentation

3.1 Installation

The program is available via a git repository [5]. To get the program just clone the repository by typing in bash:

```
1 git clone git://github.com/twesterm/ged-interface.git
```

To compile the program Qt 4.5 [6] has to be installed. There are packages available for every major Linux distribution, Mac OS X and Windows. On OpenSuse [7] libqt4-devel (shipped with OpenSuse 11.0) is required for compiling.
Enter the get-interface directory and build with:

```
1 qmake
2 make
```

A binary GED or GED.app (Mac OS X) is created which can be used.
The compile PIMAG, enter the ged-interface/PIMAG/ directory and run the compile script with:

```
1 ./compile
```

Note that the Intel FORTRAN compiler (ifort) [8] is necessary to compile PIMAG. A PIMAG binary is produced that has to be copied or installed in a central location. .

3.2 Program structure

The program is written in C++ using Qt 4.5 (documentation online: [6]).
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 are mentioned.

3.3 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 [6].

3.3.1 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.

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

Gets two points p_1 and p_2 and returns the distance between the points.

3.3.3 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 and returns the angle in the defined triangle at p_1.

3.3.4 QString GEDItem::writeInputFile()

Writes out an input file for PIMAG [2] 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.

3.4 The QMainWindow Class

This class handles the main window and all of its 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.

3.4.1 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 always false (for internal reason).

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

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

3.4.3 void MainWindowClass::on_actionOpenFile_triggered() [slot]

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

3.4.4 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.

3.4.5 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().

3.5 Porting to Windows

In general it is possible to port xpimag to windows. There are two main problems:
1.: String processing. The following memberfunctions do string manipulation and would have to be carefully modified to take the difference in file names into account:

```
1 void MainWindowClass::on_actionOpenFile_triggered()
2 void MainWindowClass::on_IntegratePushButton_pressed()
3 QString GEDItem::writeInputFile()
```

2.: The **void** MainWindowClass::on_IntegratePushButton_pressed() would have to be modified to produce scripts for Microsoft Power Shell.

3.6 Further developments

A number of features where are not yet implemented. A reasonable approach would be to implement automated rescaling of the image histogram to improve the contrast. That would make it easier see errors on the experimental data.
An other convenient feature would be the integration of PIMAG in the graphical interface. This has been tried, but failed because there was no possibility to link the C++ code of xpimag with the FORTRAN 77 of PIMAG. PIMAG does not compile with the g77 compiler and therefore fails to link with g++. A solution would be to use the Intel C++ Compiler for building and linking the whole project. If this is implemented it should be easy to implement plotting of the output files into the program.

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. Stuart Hayes for the discussions and the introduction to the topic.

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.

4 References

References

[1] Hargittai I, H. M. e. *Stereochemical Applications of Gas-Phase Electron Diffraction. Part A. The Electron Diffraction Technique.* VCH Publishers, Inc., New York, (1988).

[2] Gundersen, S., Samdal, S., Strand, T. G., and Volden, H. V. "Benzene; high level quantum chemical calculations, gas electron diffraction pattern recorded on Fuji imaging plates and a method to explore systematic discrepancies which was used to determine an improved sector correction", *JOURNAL OF MOLECULAR STRUCTURE* **832**(1-3), 164–171, APR 30 (2007).

[3] Stambulchik, E. "xmgr is a graphical plotting program". http://plasma-gate.weizmann.ac.il/Grace/ (2009).

[4] Thomas Williams, C. K. "GNUPLOT is a graphical plotting program". http://www.gnuplot.info/ (2009).

[5] Linus Torvalds, Shawn O. Pearce, J. C. H. "git is a version control software". http://www.git-scm.com (2009).

[6] Corperation, N. "Qt online documentation". http://doc.trolltech.com/4.5/index.html (2009).

[7] Cooperation, N. "OpenSuse is a Linux distribution". http://www.opensuse.org/ (2009).

[8] Cooperation, I. "Intel FORTRAN Compiler". http://software.intel.com/en-us/intel-compilers/ (2009).