

# Ramdog Manual

1.	Running Ramdog.....	1
2.	Editing the source code.....	3
2.1.	Installing the Java Development Kit.....	3
2.2.	Installing Eclipse.....	3
2.3.	Installing EGit .....	5
3.	Use Cases .....	5
3.1.	Use Case 1: Spotpicking a series of diffraction images.....	6
3.1.1.	Loading an Image .....	6
3.1.2.	Manipulate Image Color Model .....	7
3.1.3.	Input Detector Calibration .....	7
3.1.4.	Determine regions of interest.....	8
3.1.5.	Select Corresponding File Series .....	10
3.1.6.	Output File Format.....	11
3.2.	Use Case 2: Computing the Patterson Function .....	12
3.2.1.	Loading a background file. ....	13
3.2.2.	Loading the X-ray Image .....	13
3.2.3.	Loading the Calibration Data .....	13
3.2.4.	Calculating the Patterson Function.....	13
3.2.5.	Saving the Patterson Function as an image .....	14
3.2.6.	Fourier Transform a series of images .....	14
3.3.	Use Case 3: Image output .....	14
3.4.	Use Case 4: Loading calculated diffraction images.....	15
3.5.	Use Case 5: Comparing diffraction image to calculated Bragg points.....	15
4.	Spotpicking Algorithm.....	16
5.	References .....	20

## 1. Running Ramdog

Download “Ramdog.jar” from <https://github.com/ericdill/Ramdog> by clicking the “Download ZIP” button on the right hand side of the web page. Extract the zip file to a directory and double click on “Ramdog.jar” to run it. You should then be greeted with something that looks like Figure 1.

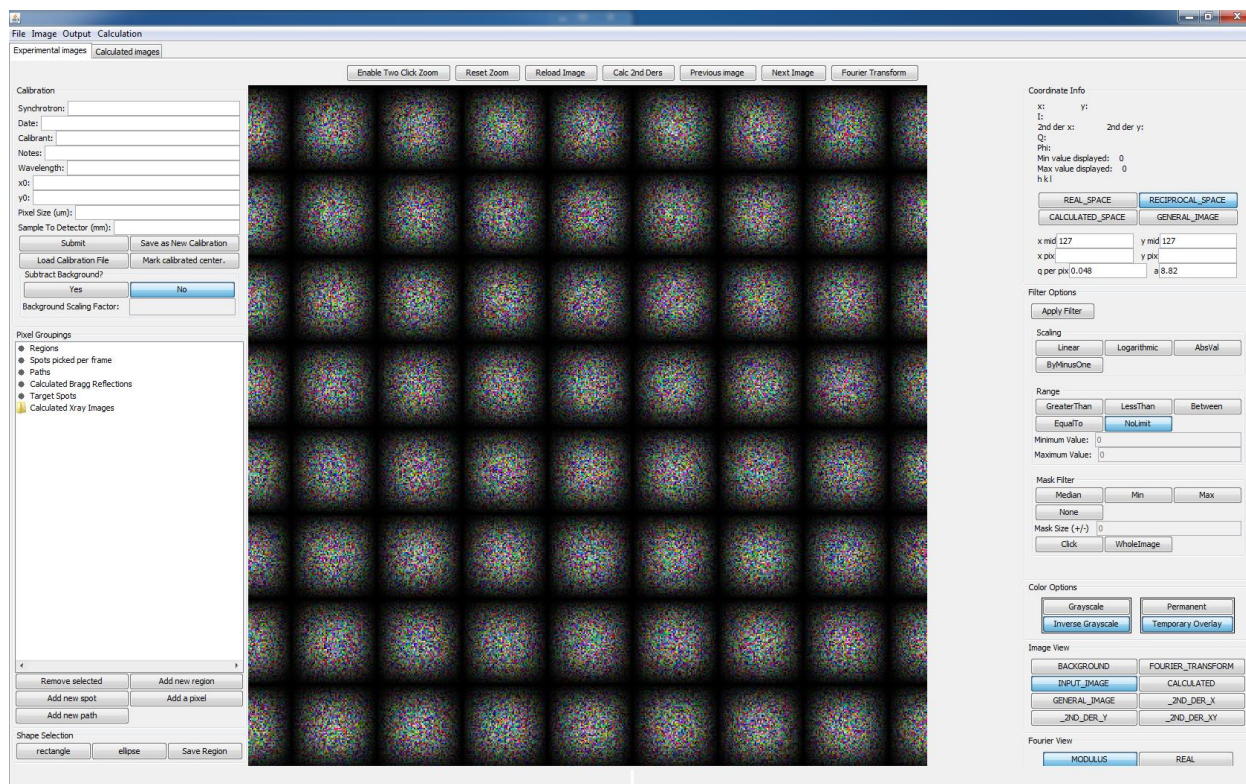


Figure 1. Screenshot of the Ramdog GUI on Windows 7.

If nothing appears it is most likely that you do not have the Java Development Kit (JDK) or Java Runtime Environment (JRE) installed. To determine whether or not these are installed, open up a command window (windows key + r, then type cmd, then press enter) or a terminal and run the command:

**java -version**

If your output looks like Figure 2, then you have a version of the JRE or JDK installed and you can skip the next paragraph.

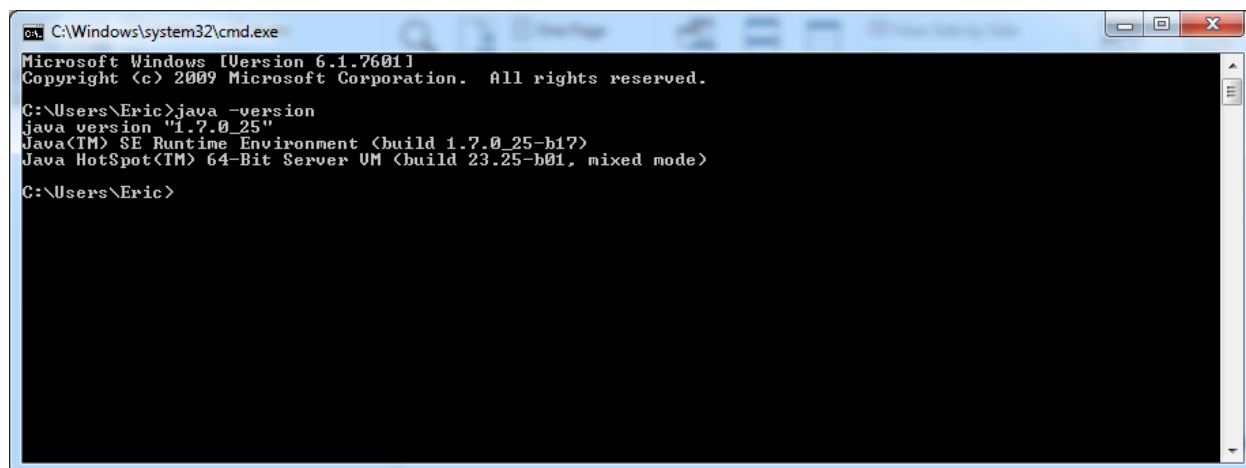


Figure 2. Verifying that the Java Development Kit (JDK) or the Java Runtime Environment (JRE) is installed.

If your output says something along the lines of “java is not recognized as an internal or external command...” then you need to install the Java Runtime Environment (JRE) or the Java Development Kit (JDK) installed. I would recommend using version 7 of either of them. Either of them can be installed from Oracle.com (or java.sun.com)

If you want to edit the source code you will need the JDK and Eclipse, otherwise the JRE is fine. See Installing the Java Development Kit on page 3. After you have installed either of these, double click Ramdog.jar and you should be greeted with Figure 1. If this still does not pop up, please rerun the java –version command and ensure that you have the JDK or JRE installed. If you still get an error, restart the computer and run the java –version command. If you’re still having issues, please get in touch with your local IT support and get java installed.

If you are positive that you have Java installed but you are not greeted with Figure 1, the next step is to run Ramdog.jar from the command line:

```
java -jar Ramdog.jar.
```

**(Protip: Shift + right click on the “Ramdog-master” folder and a new command will appear called “Open command window here.” Click that and then type in the aforementioned command.)** If you get an error while doing this, please take note of exactly what the output was and send me ([eddill@ncsu.edu](mailto:eddill@ncsu.edu)) or Prof. Jim Martin ([jdmartin@ncsu.edu](mailto:jdmartin@ncsu.edu)) an email with the error text. Please include your Java version as well (the output from the java –version command).

## 2. Editing the source code

To view the source code, navigate to <https://github.com/JamesDMartin/Ramdog/tree/master/src>. The source code was developed with Java 1.7 in the Eclipse IDE (Kepler). To edit the source code, the [Java Platform \(JDK\)](#) and the [Eclipse IDE](#) must be installed. For ease of use it is recommended to install the Eclipse extension “EGit” as the github repository can be cloned directly into Eclipse and it will be trivial to update the source code as the main repository is edited.

### 2.1. Installing the Java Development Kit

This software was developed with Java 1.7 revision 21 (1.7u21). I advise working with Java 1.7, but different versions will likely not have any issues. Download your Java Development Kit (JDK) of choice from <http://www.oracle.com/technetwork/java/javase/downloads/index.html> and then install it like a standard program.

### 2.2. Installing Eclipse

This software was developed in the Eclipse Integrated Development Environment (IDE), Kepler edition. The software repository is configured to work with the Eclipse IDE and the setup is very easy if you decide to use Eclipse. Eclipse does not need to be installed, just downloaded and extracted from the zipped file. To download Eclipse, navigate to <http://www.eclipse.org/downloads/> and choose your desired flavor. I use “Eclipse Standard” edition. Select the appropriate version (32-bit or 64-bit) and commence the download on the following page. Once downloaded, navigate to the download location and you should see a file named something like “eclipse-standard-kepler-SR1-win32-x86\_64.zip” which needs to be extracted into a folder (try right-clicking on it and look for the word “extract”). Once extracted Eclipse is operational and can be run by double-clicking on “eclipse.exe” within the extracted folder. In Windows,

I like to copy the folder into my “Program Files” directory and then create a shortcut from “eclipse.exe” to the Desktop or Taskbar, but this is not a necessary step.

Run eclipse by double clicking on “eclipse.exe” or the shortcut that you created. You will be greeted by a sequence of screens where one will ask you to “Select a workspace.” For most users the default location is fine, so click the “Use this as default and do not ask me again” checkbox and click ok. If you decide that you would like a workspace in a different location, you can change that from within the IDE after it loads (**File->Switch Workspace**).

Eclipse will now load into the “Welcome to Eclipse” screen. Feel free to navigate around and explore or just go directly to the workbench by clicking the arrow in the top right corner labeled “Workbench.”

To configure your new Eclipse install to access the Ramdog source code, you will need to install EGit. See the “Installing EGit” section for how to install this. Return to this point once you have installed EGit. Moving forward, I am assuming that you have EGit installed. If you haven’t yet done that, please go do it. It makes editing the source code and keeping it in sync with my changes extremely easy.

To add Ramdog to your Eclipse IDE, click on **File**, then **Import**, then expand “Git,” select “Projects from Git” and click “Next”. Then click “Clone URI” and click “Next.” Type “<https://github.com/JamesDMartin/Ramdog.git>” into the URI: text box. Alternatively you can navigate to <https://github.com/JamesDMartin/Ramdog> and click the ‘copy to clipboard’ icon on the right hand side of the web page and the little popup should change from “copy to clipboard” to “copied!” Once that has been copied, return to Eclipse and right click in the URI box and click paste (or Ctrl + V). After the URI box has is populated, click Next, then Next again, one more time, make sure that “Import existing projects” is selected and click next, then click Finish.

You will now need to clone another project, this time located at “<https://github.com/ericdill/GlobalPackages.git>”. Do this the exact same way that you just did added Ramdog to the workspace.

At this point you will return to the Eclipse IDE with two new folders in the “Package Explorer” window on the left hand side of the screen named something like “RamdogPrototype1 [Ramdog master]” and “GlobalPackages [GlobalPackages master]”. Expand the RamdogPrototype1 project by double clicking on the root folder or clicking the arrow to the left of the name. There is a default runnable file called “Ramdog.jar” that will run a version that I compiled. There will be a folder called “src” which contains the source code. There is also a folder named “doc” which contains this file among a few others. In the src folder there are nine package folders containing various aspects of the program source files. To run Ramdog in the Eclipse IDE navigate to the “gui” folder and double-click “**ImageDisplay.java**”. Then click the green circle with the play arrow in it (arrow in Figure 3), click “Run As” and then click “Java Application.”

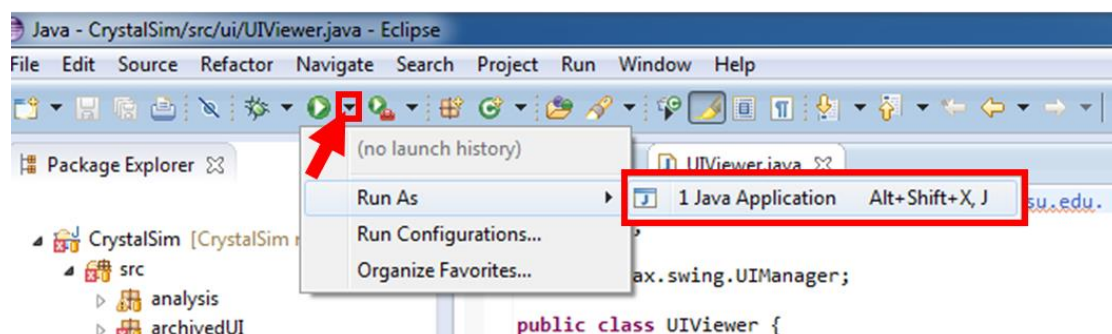


Figure 3. Running a program from the Eclipse IDE.

Because I've included various files that were useful at some point in the past but are now broken due to changes in other files, a popup will appear that is telling you that there are "Errors in the required project(s): Global Packages and Ramdog. Proceed with launch?" The errors that are in these projects do not affect the runtime behavior of Ramdog, so go ahead and "Proceed." If you so desire, you can check the box "Always launch without asking" to never see this message again. I would advise against this, but it's up to you. At this point, you should see a GUI that resembles Figure 1.

**ECLIPSE PRO TIP:** Add line numbers by right clicking just to the left of the text in the editor and click "Show Line Numbers."

**ECLIPSE PRO TIP:** Change the font size by going to Window -> Preferences. A popup will appear, go to General -> Colors and Fonts. Now in the right side of that window click on "Java" and then "Java Editor Text Font" and then "Edit" on the right hand side of the window. I like 14 point font, but I would not change the font style. Looking at source code that is not monotonically spaced is just weird.

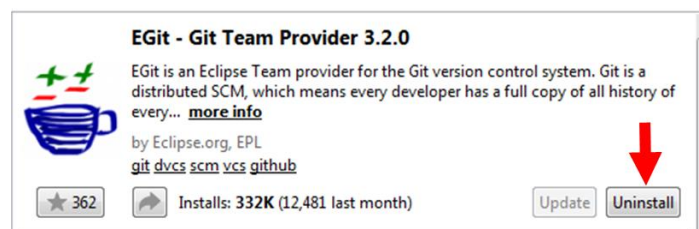


Figure 4. Click where the arrow is to install EGit from the Eclipse Marketplace. Yes, I realize that it says "Uninstall" in this image, but it will say "Install" if you have not yet installed it.

## 2.3. Installing EGit

Installing EGit is very easy from within the Eclipse IDE. Navigate to "Help" on the menu bar and then "Eclipse Marketplace." Once this loads, search for "EGit" and click the "Install" button. (I realize that my image shows an "Uninstall" button; that's because I already have it installed. If you don't have it installed your button will say "Install.") Click through the installation, accepting the various agreements. Once finished, Eclipse will require a restart. After restarting you will have EGit installed.

## 3. Use Cases

Ramdog is designed to aid in the analysis of X-ray diffraction images by providing an interface where individual pixels of the image can be easily viewed, where diffraction spots can be automatically found through a time-series of diffraction images and where diffraction images can be 2D Fourier transformed



to obtain the corresponding 2D Patterson function which is a real-space 2D pair-correlation weighted by the product of the electron density of the atom pairs. Additionally, Ramdog can load calculated diffraction images that are formatted as 3 columns (x, y, I) by N rows. Understanding how these three use-cases are implemented will explain the majority of the uses of Ramdog.

### 3.1. Use Case 1: Spotpicking a series of diffraction images

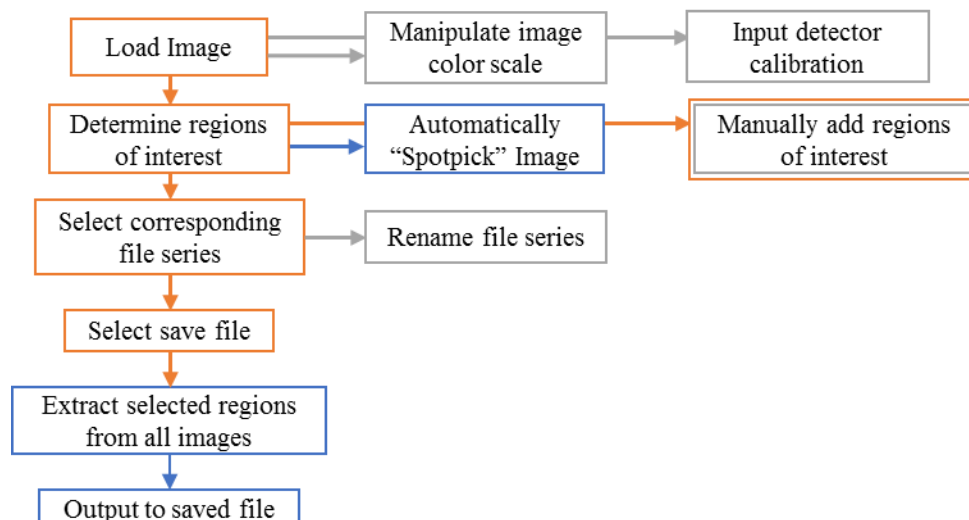


Figure 5. Flowchart of the Spotpicking use case. Orange indicates a step that requires user input. Gray indicates a step that is optional. Blue indicates a step that is automated by the software.

#### 3.1.1. Loading an Image

Ramdog supports loading xray images that are either in their native format (usually a binary file with a header and your xray data) or images that have been saved as common file formats (.png, .bmp, etc.). Loading the raw image file is preferred for a number of reasons but loading a pre-rendered image is significantly simpler. The raw image file is preferred because each pixel in the detector corresponds to one data entry while the pixels in the pre-rendered image do not necessarily correspond to one pixel in the detector. This can cause a headache when trying to relate the detector calibration info to the displayed image in Ramdog.

**Automatically determine the image color scale.** On the Menu bar under Image is a menu item called “Auto-scale Image?” which tells Ramdog to use the min and max intensity values of the loaded image to generate a greyscale image which goes from white at the min value to black at the max value. The default image view is a logarithmic scale. This can be changed with the buttons on the right-hand side of the GUI.

**Loading a pre-rendered image.** Click File->“Load image (png, jpg, bmp, etc.)”, select your image in the popup file selection window and click Open.

**Loading a raw x-ray image.** To load the raw image you need to know the image dimensions (256×256, 512×512, 2048×2048, etc.), the size of the header (if present) in bytes and the size (in bytes) of each pixel in the diffraction image. Additionally, the diffraction data, when stored in a binary format, can be little endian or big endian. If you’re unsure, choose one and see what the image looks like, then choose the other and see what it looks like. Pick the one that looks more appropriate. These options (image dimensions, header size, data element size and endianness) are options that appear when you select File-

>Load Xray Image and are required parameters for Ramdog to open the file. These fields will be automatically populated by a guess, but these are not necessarily correct.

### 3.1.2. Manipulate Image Color Model

To manipulate the mapping from xray intensity to color, click on Image->Color Model and the color model window will appear. The color model is set up as a set of intensity-color pairs which serve as the color levels. Colors are then interpolated from these color levels and displayed accordingly. There are six possible operations that the user can perform on the color model:

1. Change the intensity of the level (Value),
2. Change the color of the associated intensity level,
3. Move the intensity-color pair up,
4. Move the intensity-color pair down,
5. Remove the currently selected intensity-color pair and
6. Insert a new intensity-color pair below the currently selected level.

Operations 3-6 are performed with the buttons in Figure 6b. Operations 1-2 are performed with the snippet from the image color model window shown in Figure 6a. To change the intensity level, double click on the desired cell under the “Value” column and edit it to be the new desired value. When finished, press enter. To change the color of a level, click on the desired color level and select a new color by simply clicking on the desired color at the bottom of the window. To see the effect that your changes have on the displayed image, click the “Apply Filter” button on the main window (It’s about 1/3 of the way down the buttons on the right side of the window).

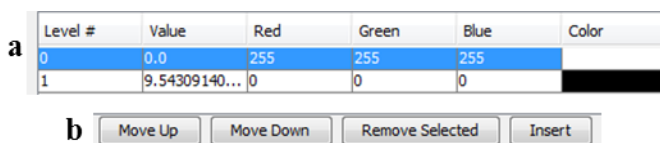


Figure 6. Relevant portions of the Image color scale window.

### 3.1.3. Input Detector Calibration

To extract any meaningful information from a diffraction image, five parameters are needed: Wavelength of the incident radiation, beam center (x,y) in pixels, the physical size of each pixel (μm) and the distance between the sample and the detector (mm). With these five parameters, the pixels in the diffraction image can be converted to Q (Å<sup>-1</sup>) according to the following three equations and further analysis becomes possible.

$$2\theta = \tan^{-1} \left[ \frac{\text{pixelSize}}{1000} * \frac{\sqrt{(x - x_0)^2 + (y - y_0)^2}}{\text{sampleDistance}} \right]$$

$$Q = \frac{4\pi \sin\left(\frac{2\theta}{2}\right)}{\lambda}$$

$$\varphi = \tan^{-1} \left( \frac{y - y_0}{x - x_0} \right) * \frac{180}{\pi}$$

The calibration section of the GUI shown in Figure 7 can be used as a temporary store for the calibration information which is not saved when the program is closed, or the calibration information, once saved, can be saved as a new calibration file which will be automatically loaded each time Ramdog starts.

To use calibration information in a temporary state simply enter the five parameters (wavelength, beam center, pixel size and sample-detector distance) and click “Submit.” After this is done, when your cursor is over the diffraction image, the coordinate info in the top right corner of the GUI will now have values for Q and phi.

To use the calibration information as a permanent state, enter the five parameters plus information about the calibration parameters (Synchrotron, Date, Calibrant and Notes). A file will then be generated called “[Synchrotron]--[Date]--[Calibrant].calib” in a folder called “Calibration” in your Eclipse directory. This file will then be automatically loaded the next time Ramdog is loaded and the parameters will be accessible from the load calibration window.

The calibration files that have been saved are accessed by clicking on the “Load Calibration File” button which opens a new window, shown in Figure 8. Each calibration file that was loaded is given a row in the table and an associated button at the top of the window. To load the new calibration file, simply click the button at the top of the window that is associated with the calibration file that you want to load. The number in the first column is associated with the buttons at the top of the window. The calibration files can be edited in this window and saved to file. The “Add new calibration file” currently does nothing and the “Delete calibration file” is a temporary delete and does not remove the file from disk.

### 3.1.4. Determine regions of interest

As the goal of this use-case is to extract information about your diffraction images as a function of time, there are a few tools available to choose these interesting regions of your diffraction image. These options are:

1. Single pixels from mouse clicks,
2. Multi-line paths based on mouse clicks,

Figure 7. Detector calibration.

File #	synchr...	date	calibrant	notes	x	y	pixel...	distance	wavel...	backg...
0	APS	2006	CeO2	default	1015.845	1004.944	200	901.089	0.13702	0.0
1	APS	2009	CeO2	default	1033.321	1020.208	200	188.672	0.13702	0.0
2	bnl	2005	none	test	800.0	787.0	100	100.0	0.922	0.0
3	NLSL	April	LaB6	19.1888	277.997	266.527	100	83.936	0.64613	0.0
4	NLSL	April	LaB6	19.1888	266.527	277.997	120	83.936	0.64613	0.0
5	NLSL	Feb_2007	LaB6	X6b	927.485	1002.781	80	215.454	0.64613	0.0
6	NLSL	June_2...	lab6_200	X7b	1170.486	1152.521	150	169.319	0.9222	0.0
7	NLSL	June_2...	lab6_400	X7b	1171.455	1151.03	150	368.795	0.9222	0.0

Figure 8. Window to load previously saved calibration files.



3. Elliptical or square regions based on two mouse clicks,
4. Find a diffraction spot in a small region around the users mouse click,
5. Find diffraction spots in the entire image.

**Manual selection of regions of interest (Options 1-4).** These options are straightforward and require the user to click on the desired option, given as a button near the bottom of Figure 9. Based on the button clicked, the set of buttons labeled “Shape Selection” in Figure 9 will change or vanish. The currently shown buttons correspond to the “Add new region” button and allow the user to select a rectangular or elliptical region of the diffraction image with a two-click scheme (i.e., clicking on opposite corners of the shape). Once the shape has been selected (it will show up as the border of a rectangle/ellipse) clicking on the “Save Region” button will save all pixels within that shape to a new Region object within the “Regions” tree structure at the top of Figure 9. Double clicking on the word “Regions” will expand it to show a new entry with some incomprehensible name along the lines of “Lanalysis.Pixel;@6e584751” which can also be double clicked on to show all the pixels that are contained within that new Region object. Clicking on any of those pixels will highlight it on the diffraction image.

The mechanism to add a path is similar to that for the addition of new regions. Clicking on “Add new path” will change the buttons under “Shape Selection” to “Close Path”, “Save Path”, “Clear Path”, and “Center Click”. To start a new path, either click “Clear Path” to remove any memory of previous clicks or just click on the diffraction image. When Ramdog registers the click, a circle will appear around where the click occurred. To add a segment to the path, click elsewhere on the image. Now a line will appear between the original click and the new click. Line segments can be added with further clicks on the diffraction image. To save this path so that Ramdog will extract these specific pixels out of each image in the time series, click “Save Path.” To close the path (join the most recent click with the first click) click “Close Path.” Finally, clicking on the button “Center Click” will move the path to the center of the image, as denoted by the currently loaded calibration. This is especially useful when radial cuts are of interest. Once the path is saved, a new entry will appear under the “Paths” heading at the top of Figure 9.

The mechanism to add a pixel was defined such that additional pixels could be added to spots, regions or paths after they had initially been chosen. To add pixels to an existing grouping of pixels, select the specific region/path/spot that you wish to add pixels to, click on “Add a pixel” and then select the pixels that you wish to add to that existing grouping of pixels. To plot the green overlay that corresponds to the pixels within that object, click on something else in that window and then click back to the region/path/spot that you wish to see the overlay for. The new pixels that you added should now be seen in the overlay.

Ramdog can automatically find spots near a mouse click by calculating the curvature in x and y (2<sup>nd</sup> derivatives in the x- and y-directions). This is more thoroughly described in §4.

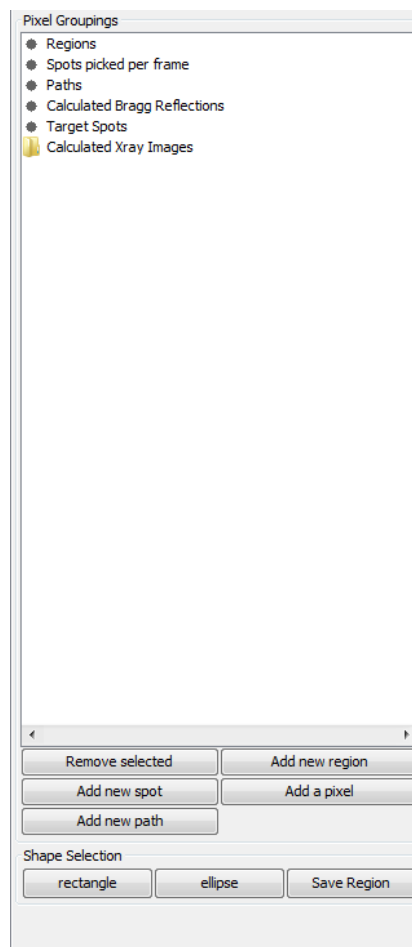


Figure 9. Pixel/Region selection portion of the GUI

**Automatic selection of Bragg diffraction spots in the entire image.** To automatically find the Bragg diffraction spots in the selected diffraction image, click on Image->“Spot pick this image” and the window shown in Figure 10 will appear. First, uncheck “Spot pick all images?”. There are a number of options that allow for a rather customizable way to set the criteria for finding the “active pixels” in the image that will then be coalesced into diffraction spots if at least some number of active pixels are touching (the default is 5). These selection criteria are (where ## represents the user input).

1. Threshold: ##
2. Pixel > ##
3. Pixel < ##
4. ## < Pixel < ##
5. Pixel == ##
6. Pixel != ##

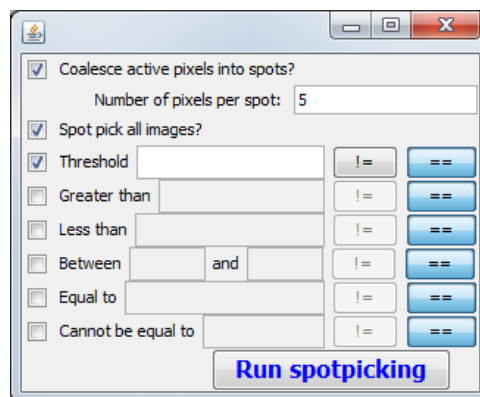


Figure 10. Spotpicking Frame.

To enable/disable these selection options click on the check box next to that option. Additionally, there are two buttons next to each of these options labeled “!= ” and “== ”. If the “!= ” button is selected, then when active pixels are found with these criteria, those pixels are rejected. When the “== ” button is selected, an active pixel found with the specific criteria is kept. Once the desired criteria have been selected, clicking the “Run Spotpicking” button at the bottom of the window shown in Figure 10 and the algorithm will process the image and save a number of spots to the Pixel Grouping called “Target Spots” shown in Figure 9. By clicking on “Target Spots” the overlay will be shown. It must be noted that this is not a perfect algorithm and the picked spots may not be exactly the same as what can be seen visually, but it is usually a pretty good approximation. If better spots are desired then perhaps playing around with the selection criterion will deliver the results you desire. Once you’re satisfied with those diffraction spots, proceed to the next step.

### 3.1.5. Select Corresponding File Series

The objective of this use case is to follow the time-evolution of pixels, regions, spots, paths, etc. At this point, you now need to tell Ramdog what the file series is. This is done by clicking Output->“Get Histories For All Regions” and then selecting all files that you wish to extract these pixel groupings from.

Once you have selected these files and clicked “Save,” a window similar to that shown in

Figure 11 will appear. **This is the order that Ramdog thinks your files belong in. If this is not the correct order, you need to rename the files such that Ramdog understands your file naming scheme.** There is an included file renaming tool that works if your files are numbered sequentially and, other than the numbering index, all parts of the file name are the same. As you can see in

Figure 11, my files are named such that the only difference is the file index which comes almost at the end of the file name. The problem here is that Ramdog sorts by the first character of the file index and does not treat the file index as a number.

To trick Ramdog into sorting the correct way, your files will be renamed to include leading zeroes into the index such that 7 becomes 007 and Ramdog sorts correctly. To do this, click on the “Rename” button and a new window will pop up asking for every part of the file name before the index in one box

and every part of the file name after the index in the second box. In the example given here, I would put “D:\Data\aps 09\eric\czxfun\raw 2D\czx1-10\_230-135-cooled\_90kev\_1s\_350\czx1-10\_230-135-cooled\_90kev\_1s\_350f.cor.” into the first box and “.cor” into the second box. Make sure to click “Set Prefix” and “Set Suffix” after placing the correct text into the box. Also note that ctrl+c and ctrl+v work here so that you do not have to type all of that text in by hand. Once you’ve entered the correct file names, click “Rename” and your files will be renamed (ON DISK, so make sure you have a backup of these files as I will not be held responsible for harming your raw data) and sorted in the correct order in

Figure 11.

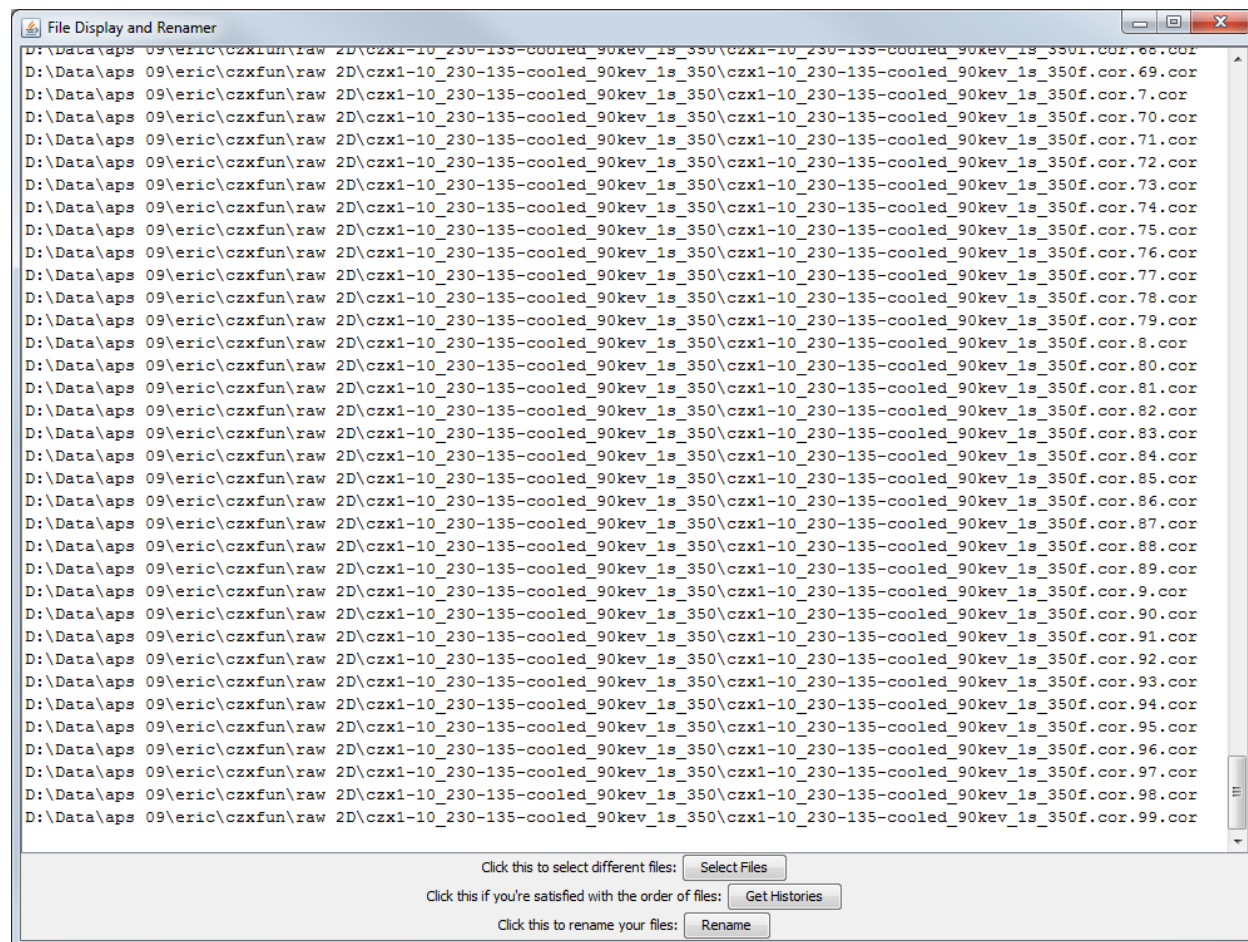


Figure 11. File series display.

After your files have been renamed, click on “Get Histories” and a save window will appear allowing you to select an output folder and a file root. Clicking on “Save” will then load each of the images, extract the relevant pixels and move on to the next image. After all images have been processed, the results will be written to disk in the specified location.

### 3.1.6. Output File Format

Multiple files will be output (where *root* is the file root that you input and *type* is region/path/spot): *root\_type.txt*, *root\_normalized\_type.txt*, *root\_QandPhiandI.txt*.

**root\_type.txt.** This file is structured so that each column corresponds to one of the diffraction paths/spots/regions of interest and the rows in each column correspond to the total intensity of the path/spot/region in each frame.

**root\_normalized\_type.txt.** This file is structured such that every column after the first corresponds to one of the paths/spots/regions of interest. The first row is the path/spot/region index. The second row is the starting intensity and the third row is the final intensity. The fourth row is Q and the fifth row is phi. Rows 6 and 7 are the average x and y coordinates of the spot/path/region. The remaining rows are the integrated and normalized (between 0 and 1) regions/spots/paths as a function of time and can be fit with the KJMA model to extract the rate constant and nucleation time for each diffraction spot.

**Root\_QandPhiandI.txt.** This file is simply a summary of the properties of the spot/path/region in the final frame. Each row corresponds to one of the spots/paths/regions of interest. The first column is Q of the spot, the second column is phi, third is the intensity in the last frame, the fourth column is the average x coordinate and the fifth column is the average y coordinate.

### 3.2. Use Case 2: Computing the Patterson Function

The Patterson function is the Fourier transform of the raw intensities of the diffraction images and gives the real-space electron density weighted pair correlations in two dimensions.

The pixels in the diffraction image, assuming a flat plate detector, do not have a constant  $\Delta Q$  which is a problem for the Fourier transform, as it expects an even coordinate grid. Thus, the raw images need to be first interpolated into a set of data with constant  $\Delta Q$ . This is performed automatically but requires the detector calibration to determine the Q-coordinates of each of the pixels and currently requires a CUDA-capable GPU to perform the Discrete Fourier Transform (DFT). An OpenCL version is on the to-do list but has not yet been finished. Note: The Fast Fourier Transform, FFT, is orders of magnitude faster but requires the data dimensions to be a multiple of two and therefore requires the data to be truncated or padded, while the DFT works on data sets of arbitrary dimensions without any extra manipulation.

Technically your measured diffraction image includes a host of factors other than the diffraction of your sample including the sample container, air scattering, the beam stop and anything else in the beam which need to be subtracted from the measured intensities before the Patterson Function is calculated. However, as many of these factors are isotropic, if your measured image is highly anisotropic (i.e., not powder rings) then the background subtraction may not be that critical. Conversely, the beam stop is highly anisotropic in your diffraction image and will present a problem unless your data is highly isotropic. Essentially, if you know your data is anisotropic/isotropic then you can ignore experimental artifacts that are the opposite (isotropic/anisotropic) since while the experimental artifacts will appear in the Patterson Function, you will be able to quickly discard them and explore the experimentally relevant pieces of the Patterson Function. Though, it must be noted that the Patterson Function must be the measured intensities less the background. All of this is to say that you may be able to get almost as much information without a background file.

Assuming that you have a detector calibration, a measured background image and a CUDA-capable GPU, then you can calculate the Patterson Function of your 2D images, as detailed below.

### 3.2.1. Loading a background file.

A background file can be loaded by clicking on the “Yes” button in the “Subtract Background?” portion of the Calibration panel, as shown in Figure 12 which will cause an open file dialog to appear. To load your raw background file you will need to know the image dimensions, data element size and the file header size. Enter those values into the corresponding text boxes and clicking “Open” will result in your background image appearing in the image display window. The background subtraction is scaled based on the text box labeled “Background Scaling Factor.” To switch between the background and the diffraction image, click on the buttons labeled “BACKGROUND” or “INPUT\_IMAGE”, respectively, as shown in Figure 13 and then click “Apply Filter” on the right side of the UI.

### 3.2.2. Loading the X-ray Image

See “Loading an Image” (§3.1.1 on page 6).

### 3.2.3. Loading the Calibration Data

See “Input Detector Calibration” (§3.1.3 on page 7).

### 3.2.4. Calculating the Patterson Function

Before the Patterson Function can be calculated, you need to tell Ramdog what your desired  $\Delta Q$  is. This is done by entering the desired value into the “q per pix” text box in the top right corner of the GUI in the “Coordinate Info” section (Figure 14). By clicking the “Fourier Transform” button (located in the buttons along the diffraction image), Ramdog compute the Discrete Fourier Transform of your 2D diffraction image which has been linearly interpolated onto a grid whose step size is defined by the entered value of  $\Delta Q$ . After the Fourier Transform is complete, the new image will be visible, though the color model will probably need to be adjusted (See Manipulate Image Color Model, §3.1.2, page 7). To switch between the Patterson Function, the original image and the background image simply click on the corresponding button, as shown in Figure 13, and then click “Apply Filter” on the right side of the UI. You will also need to update the coordinate info by clicking on the appropriate buttons (Figure 14) so that Ramdog can calculate the coordinates of your mouse cursor according to the proper coordinate space.

Finally, the Patterson Function can be viewed in four ways, shown in Figure 15. The real and imaginary buttons simply show the real and imaginary parts of the Patterson Function, the modulus is  $\sqrt{re^2 + im^2}$  and the power spectrum is  $re^2 + im^2$ . After clicking on the desired button, click “Apply Filter”. You may need to adjust the color model after changing the Fourier view.

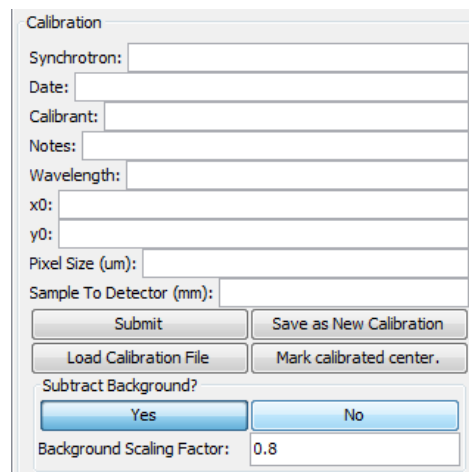


Figure 12. Loading the calibration file.

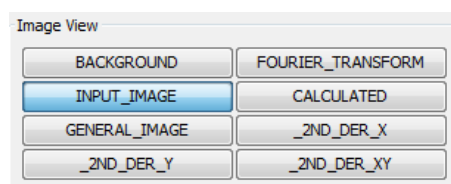


Figure 13. Loading the calibration file.

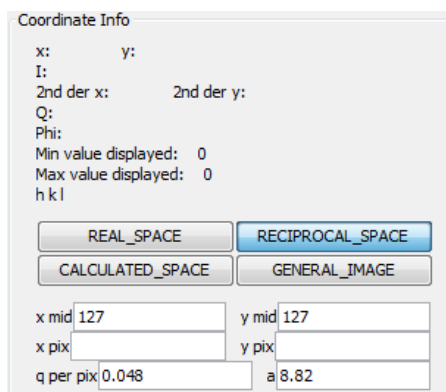


Figure 14. Coordinate Info Panel.

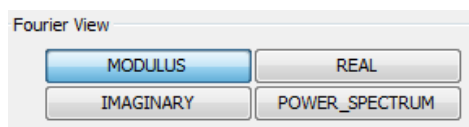


Figure 15. Fourier view options.

### 3.2.5. Saving the Patterson Function as an image

See §3.3 on page 14.

### 3.2.6. Fourier Transform a series of images

Ramdog has the capability to automatically compute the Patterson Function for a series of diffraction images and save the result in a png formatted image to disk. This is done with the menu bar by clicking “Output”->“Fourier Transform Series”. An open dialog will appear asking you to select the files you wish to automatically Fourier Transform. Once you select your images and click “Open” the files will be automatically loaded, saved to disk as an image with the extension “-auto.png”, Fourier Transformed and saved to disk as an image with the extension “--FT-raw.png”. It would probably be beneficial to load your raw image, set the color levels that you want and then Fourier transform the image and set the color levels for that as well. If you do this then your output images will be correctly formatted.

### 3.3. Use Case 3: Image output

Click on “Output” on the menu bar and then click “Save Image”. The window shown in Figure 16 will appear which provides flexible options for saving your image. The three buttons along the top, “Select Entire Image”, “Select Specific Region” and “Select Region From Image Center” control the mechanism by which images are saved.

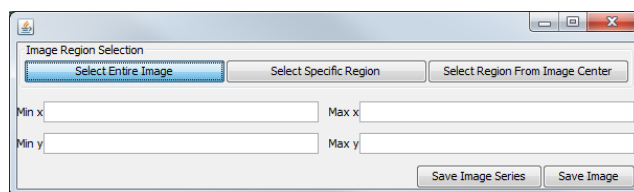


Figure 16. Image Saving Option Window.

**Select Entire Image.** This option simply saves the entire image in png format based on the Java ImageIO output standard.

**Select Specific Region.** This option allows the user to input a set of pixel coordinates to save only a specific region of the image to disk. Image is saved in png format.

**Select Region From Image Center.** This option allows the user to select a region of the image based on the center of the image. Image is saved in png format.

**Save Image Series.** The “Save Image Series” button pops up a file open dialog that asks for the series of raw diffraction images that are to be opened and saved to disk as images in png format. You will need to set the color model (“Image”-> “Color Model”) for your images before you try to automatically save them to disk unless you have selected the auto-scaling option (“Image”->“Auto-scale Image?”). The automatically generated images will be saved according to the options selected in Figure 16.



### 3.4. Use Case 4: Loading calculated diffraction images

Diffraction images that were calculated with PlasticSim (<http://www.github.com/ericdill/PlasticSim>) can be natively loaded with RamDog with “File”->“Load calculated xray images” which will show a file open dialog asking you to select a calculated diffraction image. Ramdog will automatically parse the other files that are in the same folder as the file that you just loaded to find files that contain the character sequences [100], [110] and [111]. Ramdog will automatically place those in the appropriate tree structure shown in Figure 17. All other files will be placed under the “other” node in the tree

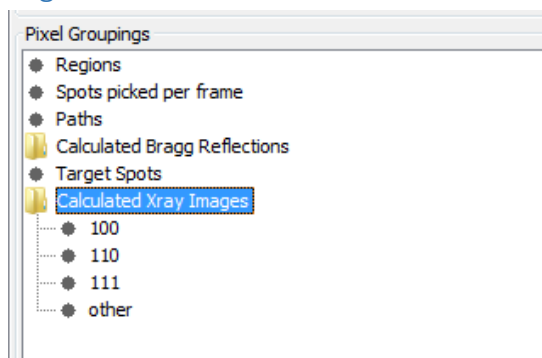


Figure 17. Calculated xray images

structure in Figure 17. By clicking on any of the calculated xray images that have now been loaded into Ramdog, the calculated image will appear in the image display portion of the UI. This image can now be manipulated identically to the experimental diffraction images and can be Fourier Transformed to show the associated Patterson Function. Additionally, the Coordinate Info panel (Figure 14) will show the appropriate x/y/l/Q/phi values if the “CALCULATED\_SPACE” button is selected. The header of the images calculated with the PlasticSim software contains the diffraction axes and  $\Delta Q$  which Ramdog uses to determine the Q value of each of the pixels in the diffraction image. These calculated images can be saved as png images by following the instructions in §3.3 on page 14.

### 3.5. Use Case 5: Comparing diffraction image to calculated Bragg points/rings

To access the Bragg point/ring window, click on “Calculation”->“Setup Bragg Reflection Calculation” and the window shown in Figure 18 will appear. This window allows you to calculate reciprocal lattice points based on a set of atoms in crystal coordinates. These reciprocal lattice points can then be plotted on top of your input diffraction image. See §3.5.2 on page 16 for an explanation of this option. Alternatively, you can give Ramdog a file that contains a set of information in five tab-delimited columns (Q, h, k, l and intensity). These rings can then be overlaid onto a diffraction image. See §3.5.1 on page 15.

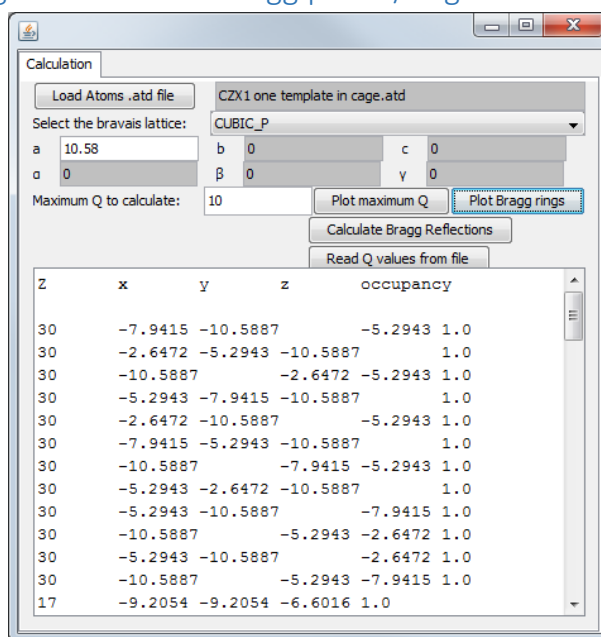


Figure 18. Bragg reflection calculation setup window.

#### 3.5.1. Plotting Bragg Rings

To overlay the Bragg rings onto a diffraction image, you need to provide Ramdog with calibration information so that it can translate pixels to Q and a tab-delimited file with five columns of data in the order Q, h, k, l and intensity. The intensity column can be filled with zeroes, but it still needs to be there. The (h, k, l) columns are so that when you mouse over a ring, Ramdog can translate that ring into hkl values as opposed to just Q. To load this information into Ramdog, click the “Read Q values from file” button and provide Ramdog with the file that contains this information, which, if the file is set up

correctly will appear in the text area in Figure 18 once Ramdog has parsed it. After these values have been successfully parsed by Ramdog, clicking on the “Plot Bragg rings” button will overlay rings at the specified Q values onto your diffraction image. Additionally, if you click on the “Permanent” toggle button on the right side of the UI then the Bragg rings will become part of the image and will not disappear when you change the zoom level of the image.

### 3.5.2. Plotting Bragg Points

**NOTE:** This feature is still in development and is only barely functional.

We frequently use the program “[Atoms](#)” to view our crystal structures. Atoms can output the atoms in the crystal in crystal coordinates (i.e., normalized to between 0 and 1) to an .atd file. This file is the target for reading in the crystal coordinates into Ramdog. To load one of these .atd files, click the button “Load Atoms .atd file” and an open dialog will appear. Select your .atd file and Ramdog will parse the atom coordinates and present its parsed information in the text box as shown in Figure 18. Once you tell Ramdog what the unit cell parameters are ( $a$ ,  $b$ ,  $c$ ,  $\alpha$ ,  $\beta$ ,  $\gamma$ ) then you can click on “Calculate Bragg Reflections” and Ramdog will use those atoms and unit cell parameters to calculate all Bragg points out to the value given in “Maximum Q to calculate”. Once this calculation is complete, a number of “Calculated Bragg Reflections” will appear in the “Pixel Groupings” portion of the UI (Figure 9). Clicking on the “Calculated Bragg Reflections” node will overlay the Bragg points oriented along one of the [001] directions. The eventual goal is to be able to rotate these Bragg points such that they can be aligned with the Bragg peaks in the diffraction image.

## 4. Spotpicking Algorithm

To analyze the nucleation and growth of individual crystallites from the 2D TtXRD data, the “active” pixels that correspond to diffraction spots must be located and extracted from each of the images in the experimental time series. It was necessary to develop an algorithm to automatically find diffraction spots in 2D images and correlate them through time as it would be prohibitively time consuming to analyze each set of diffraction images by hand. However, differentiating the signal of a crystallite from the amorphous background is challenging. To locate diffraction spots, we recognized that the first and second derivatives can be used to locate various points of a peak. The first derivative corresponds to the slope and curvature of the original peak which are zero at local maxima and inflection points, respectively, as shown in Figure 19 for a Gaussian peak ( $A=1$ ,  $x_0=3$ ,  $\sigma=0.3$ ). The position where the first derivative is zero could theoretically be used to determine the peak positions but due to the presence of noise in the collected data, detecting zero is often extremely cumbersome in practice. The problem is further exacerbated because each derivative taken reduces the signal to noise ratio. However, as the 2<sup>nd</sup> derivative is negative between the inflections points and strongly so at the center of the peak in the original function, as shown in Figure 19 using the 2<sup>nd</sup> derivative as opposed to the 1<sup>st</sup> derivative can circumvent the detection-of-zero problem as detecting signal is considerably simpler.

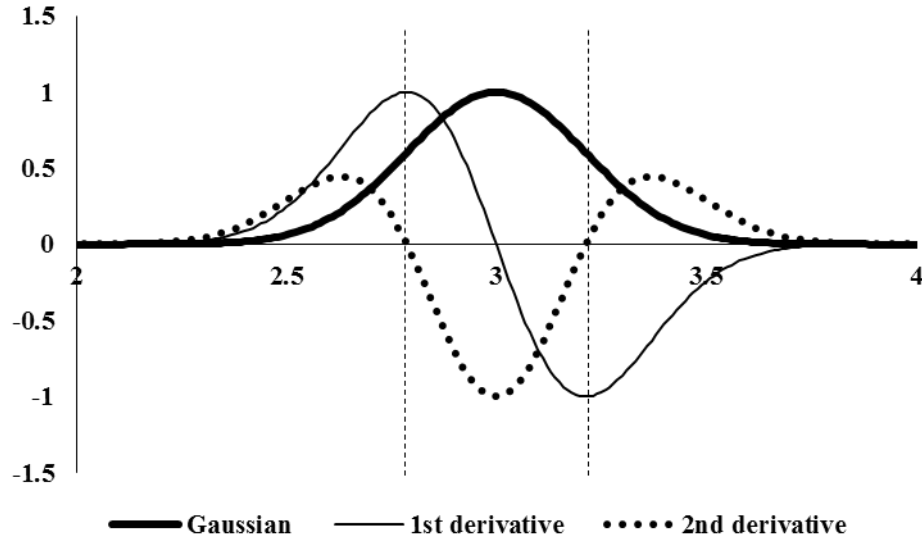


Figure 19. A Gaussian peak ( $A=1$ ,  $x_0=3$ ,  $\sigma=0.3$ ) and its 1<sup>st</sup> and 2<sup>nd</sup> derivatives. Vertical lines indicate the position along  $x$  where the 2<sup>nd</sup> derivative is zero.

Derivatives can be analytically calculated only if the original function is known, i.e.  $f''(x) = \frac{d^2}{dx^2}f(x)$ . If the original function is not known then derivatives must be calculated numerically, for which many methods exist. We have chosen to use the method of Savitsky and Golay,<sup>4.1</sup> generalized to two dimensions, where local linear regression is used to determine the coefficients for an  $n^{\text{th}}$  order polynomial. This method works by taking the inner product of the data and a “mask” which corresponds to a specific numerical manipulation like the first derivative, second derivative, mixed derivative, etc. The mechanism by which masks are constructed is not necessarily intuitive, necessitating a short description.

To compute the second derivative at each pixel in the 2D diffraction images, such as the one shown in Figure 20a, a 4<sup>th</sup> order polynomial in  $x$  and  $y$  was used (Equation 4.1) with a  $7 \times 7$  mask, where the 2<sup>nd</sup> derivative is computed for the pixel at the center of the mask ( $x_c, y_c$ ). The derivation of our 2<sup>nd</sup> derivative masks will therefore be presented with those parameters.

$$\begin{aligned}
 f(x, y) = & \mathbf{a}_0 + \mathbf{b}_0x + \mathbf{b}_1y + \mathbf{c}_0x^2 + \mathbf{c}_1xy + \mathbf{c}_2y^2 + \mathbf{d}_0x^3 + \mathbf{d}_1x^2y \\
 & + \mathbf{d}_2xy^2 + \mathbf{d}_3y^3 + \mathbf{e}_0x^4 + \mathbf{e}_1x^3y + \mathbf{e}_2x^2y^2 \\
 & + \mathbf{e}_3xy^3 + \mathbf{e}_4y^4
 \end{aligned}
 \tag{4.1}$$

$$\mathbf{M}_{xy} = \begin{bmatrix} x & y & x^2 & xy & y^2 & x^3 & x^2y & \dots & y^4 \\ a_0 & b_0 & b_1 & c_0 & c_1 & c_2 & d_1 & d_2 & \dots & e_4 \\ 1 & -3 & -3 & 9 & 9 & 9 & -27 & -27 & \dots & 81 \\ 1 & -2 & -3 & 4 & 6 & 9 & -16 & -18 & \dots & 81 \\ 1 & -1 & -3 & 1 & 3 & 9 & -1 & -9 & \dots & 81 \\ 1 & 0 & -3 & 0 & 0 & 9 & 0 & 0 & \dots & 81 \\ 1 & 1 & -3 & 1 & -3 & 9 & 1 & 9 & \dots & 81 \\ 1 & 2 & -3 & 4 & -6 & 9 & 16 & 18 & \dots & 81 \\ 1 & 3 & -3 & 9 & -9 & 9 & 27 & 27 & \dots & 81 \\ 1 & -3 & -2 & 9 & 6 & 4 & -27 & -12 & \dots & 16 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\ 1 & 3 & 3 & 9 & 9 & 9 & 27 & 27 & & 81 \end{bmatrix} \quad 4.2$$

$$\mathbf{A}_{xy} = (\mathbf{M}_{xy}^T \mathbf{M}_{xy})^{-1} \mathbf{M}_{xy}^T \quad 4.3$$

Determining the “mask” for the 2<sup>nd</sup> derivatives in x and y requires constructing the matrix  $\mathbf{M}_{xy}$ , as shown in Equation .2 and transforming it into the linear regression coefficients,  $\mathbf{A}_{xy}$ , with Equation 4.3. Each row in the resultant linear regression coefficient matrix is associated with a specific operation. Given the specifics of the construction of  $\mathbf{M}_{xy}$ , the first six rows in  $\mathbf{A}_{xy}$  correspond to the smoothing operation, the first derivative in x, the first derivative in y, the second derivative in x, the mixed derivative  $\frac{d^2}{dxdy}$  and the second derivative in y, respectively. The numerical values of these six masks are presented in Figure 19.

A 7×7 subsection of the 2D diffraction image, centered on the pixel which the 2<sup>nd</sup> derivatives in x and y are to be calculated ( $x_c, y_c$ ), is then extracted and converted to a vector such that the (x,y) coordinates of the pixel and mask are in correspondence (see the columns labeled “x” and “y” in Equation .2). Finally, the 2<sup>nd</sup> derivatives in x and y at ( $x_c, y_c$ ) can be determined by computing the inner product between rows 4 and 6 of the  $\mathbf{A}_{xy}$  4 and 6 4 and 6 4 and 6 matrix and the data vector. The results of computing the 2<sup>nd</sup> derivatives in y and x are shown in Figure 20b-c, respectively and further discussed below.

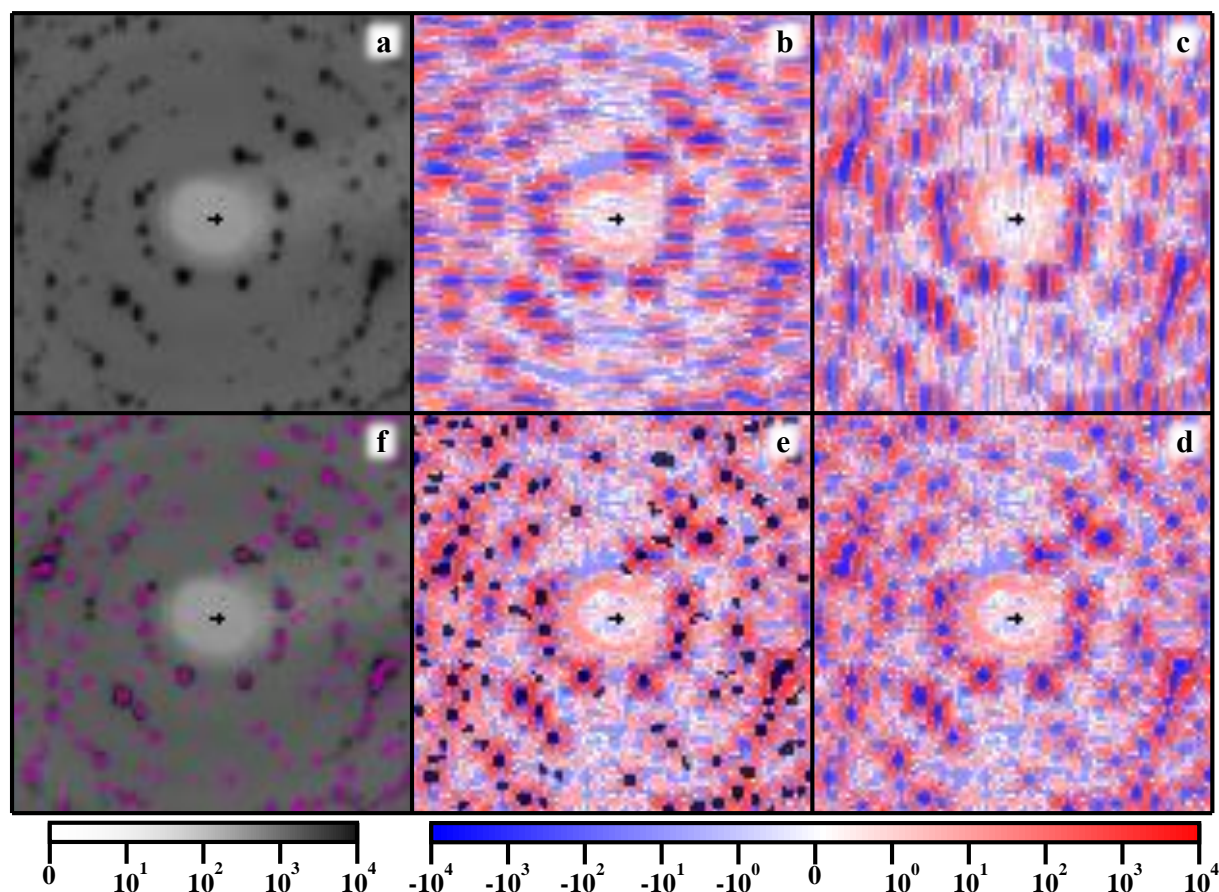


Figure 20. Isothermal crystallization of CZX-1 (145 °C). Black cross, calibrated image center. **(a)** Raw image. **(b)**  $d^2/dy^2$ . **(c)**  $d^2/dx^2$ . **(d)**  $d^2/dy^2 + d^2/dx^2$ . **(e)** Black, spots found by the algorithm described in the text in black. **(f)** Purple, spots found by the algorithm on the original image.

To visually present the algorithm for finding pixels that comprise Bragg spots, consider the oligocrystalline diffraction image that resulted from an isothermal crystallization experiment carried out at 145 °C, shown in Figure 20a. In this image, the intensity increases from white to black such that Bragg diffraction is black. The cross in the center of the image is the location of the beam center, as determined by CeO<sub>2</sub> and LaB<sub>6</sub> calibrants. Shown in Figure 20b-c are the 2<sup>nd</sup> derivatives in the y- and x-directions, respectively, colored such that blue is negative, white is zero and red is positive. It is immediately apparent that the Bragg diffraction spots appear as red-blue-red sequences relative to the directionality of the 2<sup>nd</sup> derivative. The sum of the 2<sup>nd</sup> derivatives is shown in Figure 20d where the location of the diffraction spots is appear as blue circles surrounded by a red ring. Pixels that are less than an empirical “threshold” value in both directions of the 2<sup>nd</sup> derivatives are marked as “active” and therefore potentially part of a diffraction spot. The empirical intensity threshold value depends on the signal-to-noise ratio and is therefore dependent upon the synchrotron flux, exposure duration, detector characteristics, chemical composition of the sample, sample thickness, etc. It was found that a threshold value of -5 was effective at excluding much of the noise at the expense of very little of the diffraction spots, though the exact value is fairly flexible.

Our criterion for determining if active pixels become part of a “spot” is that more than five “active” pixels must be in direct contact along the x- or y-direction. The spot size parameter serves to exclude random active pixels, differentiating them from actual crystallites. Collections of five or more

neighboring “active” pixels was determined to be an effective minimum size threshold for a diffraction spot. The actual size of the diffraction spot on the detector generally depends on the crystallinity of the sample, sample-to-detector distance and the pixel density. The image shown in Figure 20e has been overlaid with the spots found by this method with the threshold value set to -5 and the number of active pixels per spot set to 5.

It is apparent that the majority of experimentally observed diffraction spots have a corresponding algorithmically determined diffraction spot. Finally, the algorithmically determined Bragg spots are overlaid onto the experimentally obtained diffraction image in Figure 20f, demonstrating that this approach, based on a set of rudimentary criteria, successfully finds the majority of the diffraction spots. This process is called “spotpicking.”

The intensity of the algorithmic diffraction spots,  $I_{tot}$ , can be obtained by summing the corresponding pixels. As shown in Figure 20f, many of the experimentally observed diffraction spots are still visible in black behind the algorithmically determined spots, indicating that the algorithm, while successfully locating the majority of the Bragg diffraction, is not as effective at encompassing all diffraction from all spots. Additionally, the efficacy seems to be related to the size of the experimental spot as the smaller spots appear to be better “hidden” by the purple overlay. As such, it is likely that  $I_{tot}$  for the small spots will provide a better quantitative agreement with the experimental intensity.

The center of each diffraction spot can be determined by taking an intensity-weighted average of the spot’s pixels, as described in Equation 4.4, which is an effective measure of the (x,y) position of the spot since the discrepancy between algorithm and experiment is greatest at the lower intensity edge of the spot.

$$\sum_{p=1}^{all\ pixels} \frac{I_p(x_p, y_p)}{I_{tot}} \quad 4.4$$

## 5. References