Polo Documentation

Release 0.0.1

Ethan T. Holleman

Jun 23, 2020

CONTENTS:

1 About 1

2 Polo Quickstart Guide 3

3 FAQs 5

4 User's Guide 7

5 Beta Testers Guide 29

6 polo package 31

7 Indices and tables 37

Python Module Index 39

CHAPTER

ONE

ABOUT

1.1 Background

One of the largest hurtles to obtaining X-ray diffraction data from biological samples is growing large, high quality crystals.

Currently, there is not way to reliably predict successful crystallization conditions based on protein sequence alone and so high-throughput approaches are very appealing. High-throughput crystallization screens test a large chemical space using hundreds of different crystallization cocktails at the nano-drop scale. Successful conditions can then be scaled up and optimized to grow larger crystals.

The high-throughput crystallization screening center at the Hauptman-Woodward Medical Research Institute provides this high-throughput screening service to users; offering 1536 condition screens for both soluble and membrane protein samples. Each plate is imaged over a period of two months in using both visible light microscopy and UV-TPEF photography.

This high-throughput produces a large volume of images that must be sorted through in order to pick out the best condition; a task that can be very tedious and repetitious.

In 2019 Bruno *et al* published Classification of crystallization outcomes using deep convolutional neural networks which included a CNN model that could accurately classify crystallization screening images, opening the door to automating this process. The MARCO model has been used in large scale projects such as the xtution database but has not been utilized in a average-user oriented graphical program.

Polo is therefore designed to incorporate the benefits of the MARCO model and integrate the functionality of established crystallization image labeling software such as MacroscopeJ to create a GUI targeted for HWI and other high-throughput crystallization screening users that incorporates all the tools needed to go from raw crystallization images to designing optimization screens without the need to install any dependencies.

CHAPTER

TWO

POLO QUICKSTART GUIDE

2.1 Windows

Do this for Windows machines

2.2 Linux

Do this for linux (Ubuntu)

CHAPTER

THREE

FAQS

A place to include frequently asked questions

CHAPTER

FOUR

USER'S GUIDE

If you have yet to install Polo on your machine, head to the installation guide here.

4.1 Importing and Opening Image Data

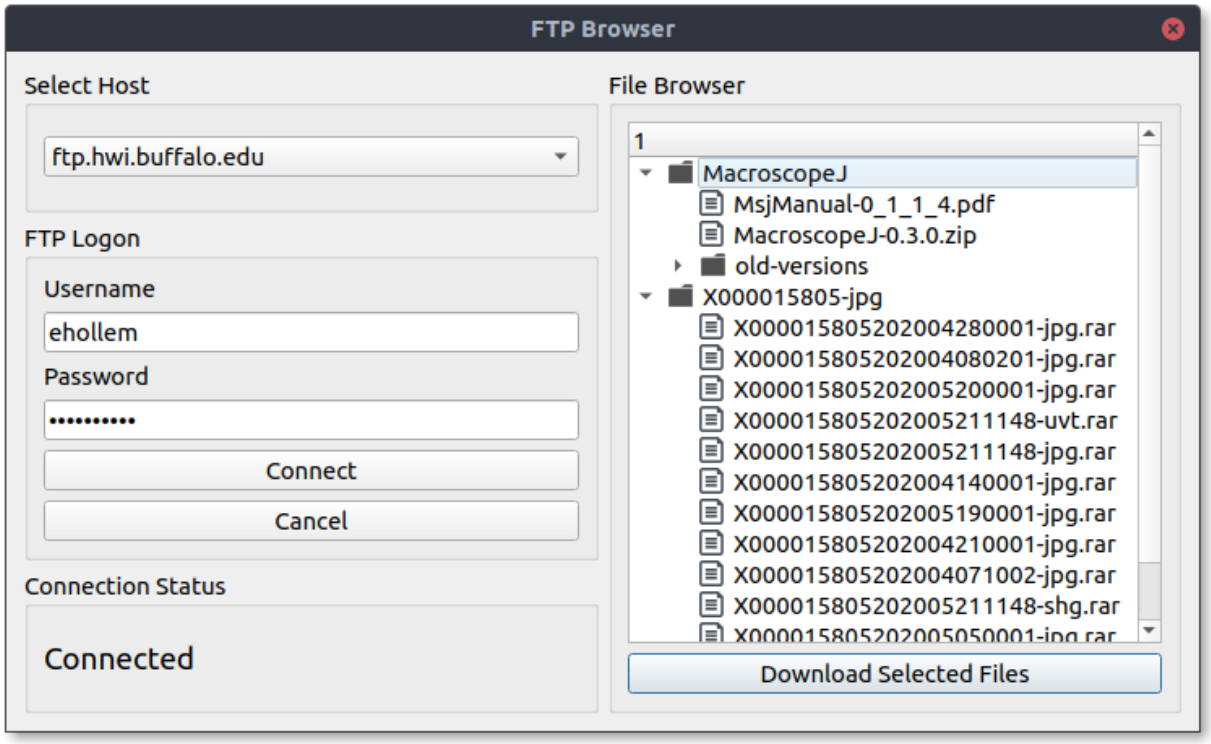
The first step to using Polo is adding your own data. Polo organizes data into "runs", which consist of a set of related screening images. Polo organizes runs into three different categories which are described below.

- HWI Screening Runs
- Non-HWI Screening Runs
- Raw Image Collections

4.1.1 Getting Your Data (via FTP)

If you do not already have your data downloaded to your local machine, Polo includes an FTP file browser which utilizes Python's ftplib package. For HWI users this allows you to download your screening images from the HWI server without leaving the application.

To open the FTP file browser, on the menubar navigate to Import -> Images -> From FTP. Enter your credentials in the new window. Once you are connected to a server files available to download will be listed in the browser menu.

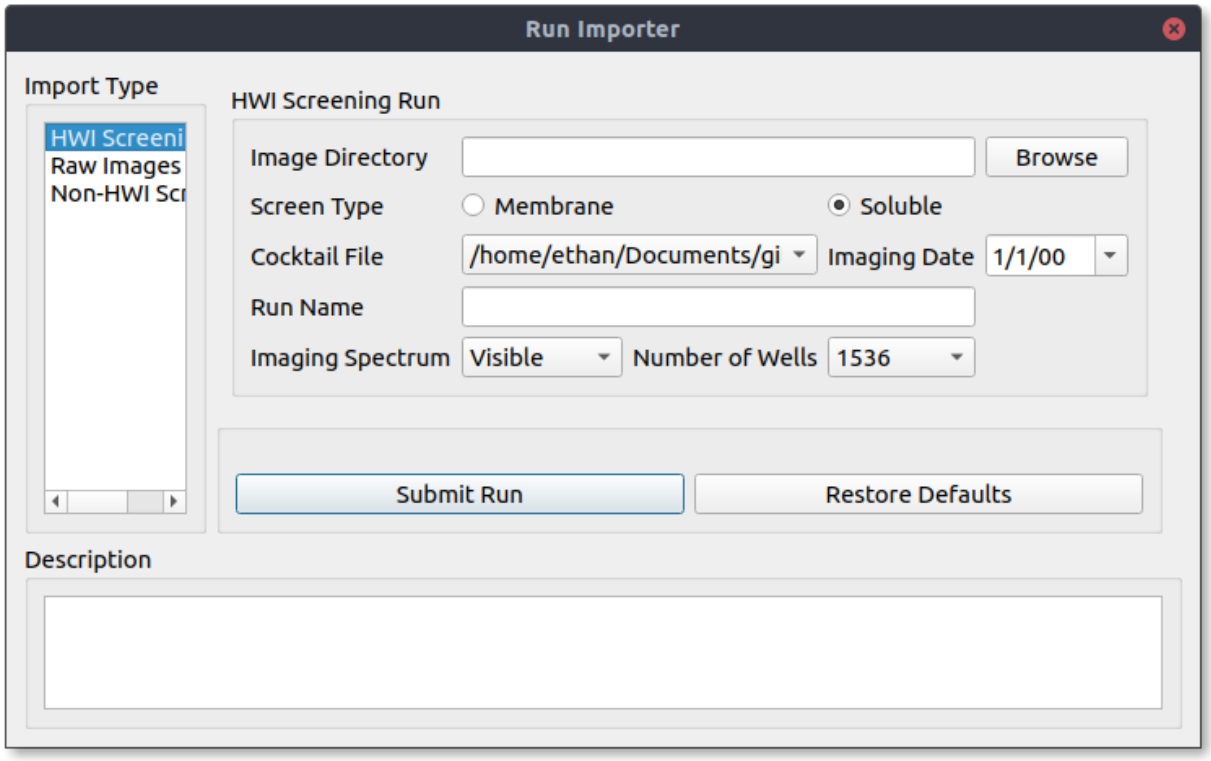


Select the checkboxes next to the files you wish to download, or select all files in a directory at once. Pressing "Download Selected Files" will start the download.

Since these are large files, Polo will download them in the background so you can continue to the program normally. Once all files are downloaded Polo will notify you that your download is complete. Closing Polo before this will result in an incomplete download.

4.1.2 Importing Images from a Directory

Once you have your images on your local machine you can import them into Polo by using the Run Importer tool. It can be opened by navigating to the menu bar and selecting Import -> Images -> From Directory. A window like the one below will then open.



If you are importing HWI screening images select **HWI Screening Run** from the **Import Type** window (Beta testers do this). Select **Browse** to open a file browser and select the directory containing the images you want to import.

Other import types are included for compatibility with other high-throughput protocols but metadata for these import modes is out of the scope of the program. This means runs imported as these types will have fewer available features.

However, since HWI has standard naming conventions Polo will suggest settings for your run including what cocktail file to use, the date of imaging, the spectrum and the number of wells. However, if you want to change any of these settings you are free to do so.

Once you are happy with your import click **Submit Run** to load your images into Polo.

4.1.3 Importing a Saved Run

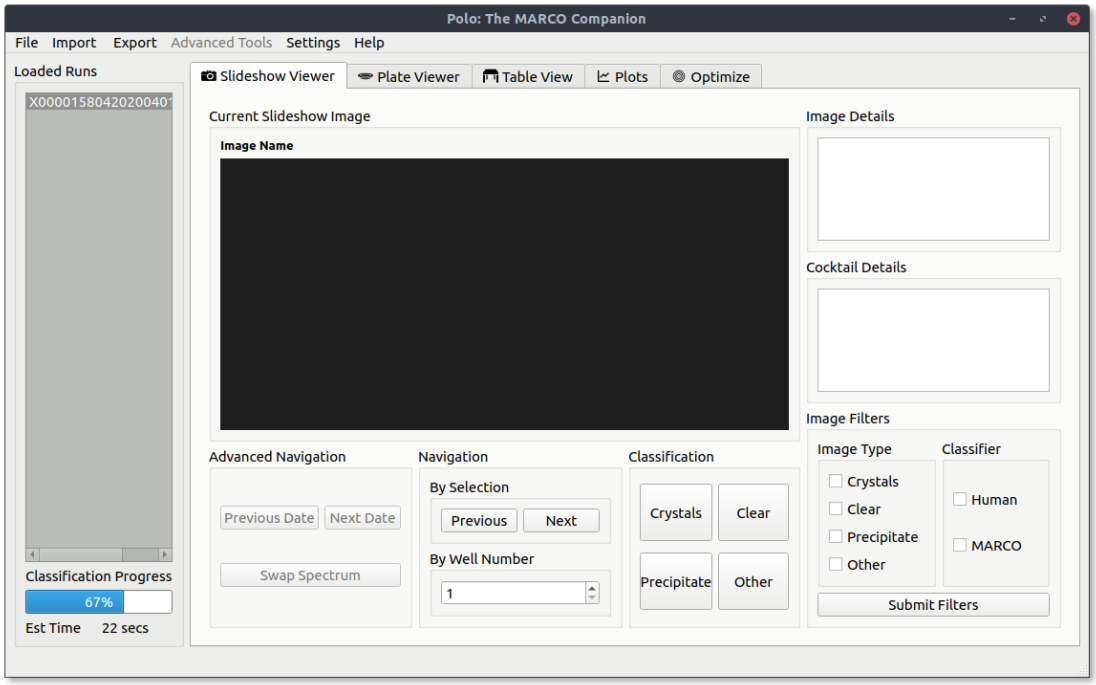
One of the advantages of Polo is the introduction of the .xtal file format. Xtal is a json like file that can store all the data relating to an individual screening run in a single file with no other dependencies. Xtal files store images, classifications, cocktail data and other annotations made while using the Polo program.

If you have xtal file ready for import you can load it into Polo by navigating to the menu bar and selecting **Import** -> **Images** -> **From Saved Run**. This will open a file browser and allow you to select the xtal file you wish to import.

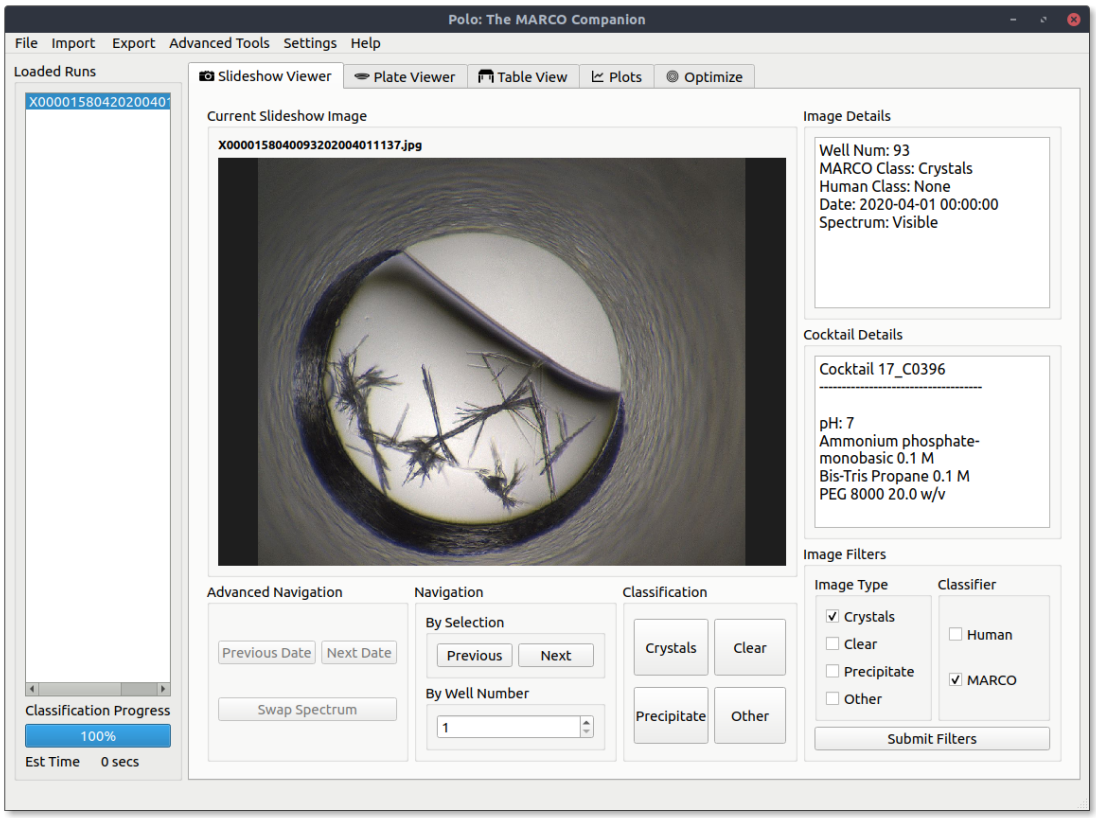
4.1.4 Opening a Run

Once a run has been successfully imported, the run name will appear in the **Loaded Runs** list, like in the image below.

If it is a new run double clicking on the run name will run the MARCO model on the run's images. You can check on MARCO's progress by using the **Classification Progress** bar located just below the Loaded Runs list. Polo will also attempt to estimate the time remaining in your classification job.

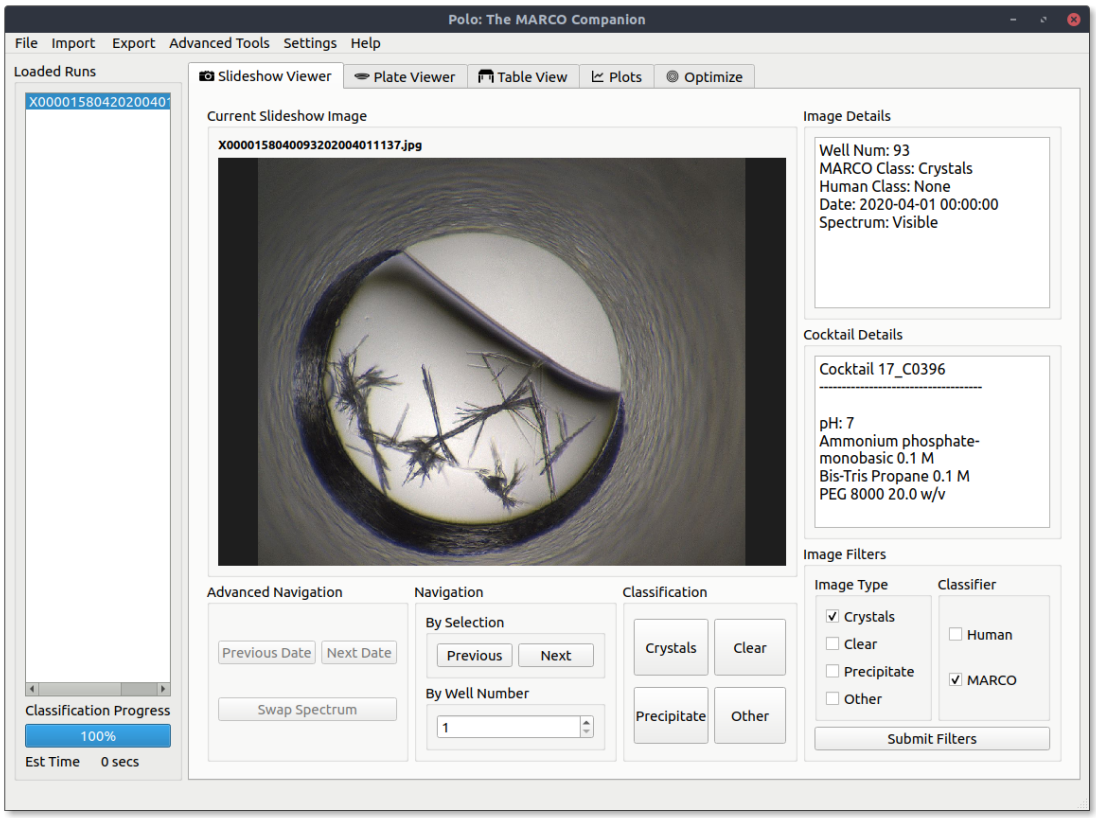


Once a run has been classified you can load it into the current view by double clicking on it again. This will set the selected run as your **Current Run** and all actions will be taken in reference to this run. You can change your current run by double clicking another loaded run in the **Loaded Runs** tab.



4.2 Using the Slideshow View

The slideshow view is the main Polo user interface. It allows you to view your screening images, label them and filter them by MARCO classifications, your own classifications or both.



4.2.1 Basic Navigation

Once you have images loaded in you can cycle through them by pressing the **Next** or **Previous** image buttons under the navigation panel. You can also use the right or left arrowkeys respectively. If you are viewing an HWI screening run you can also navigate directly to a specific well number by entering the well number into the **By Well Number** box.

4.2.2 Classification

Arguably the most important Polo feature is the ability to easily label your screening images. To label the currently displayed image press the button in the **Classification** panel with your desired label. You can classify images as Crystals, Precipitate, Clear or Other. To increase your speed you can classify images using keyboard shortcuts which are listed below.

- 1: Crystal
- 2: Precipitate
- 3: Clear
- 4: Other

Classifying an image will automatically move you to the next image in the slideshow.

4.2.3 Filtering

Using the checkboxes under the **Image Filters** panel in the lower right corner of the window will allow you to filter the kinds of images in your current slideshow. For example if you only wanted to see images that MARCO has classified as Crystal you could check the Crystal box under **Image Types** and MARCO under the **Classifier** panel. If you had checked Human instead only images that you have classified as Crystal would be shown.

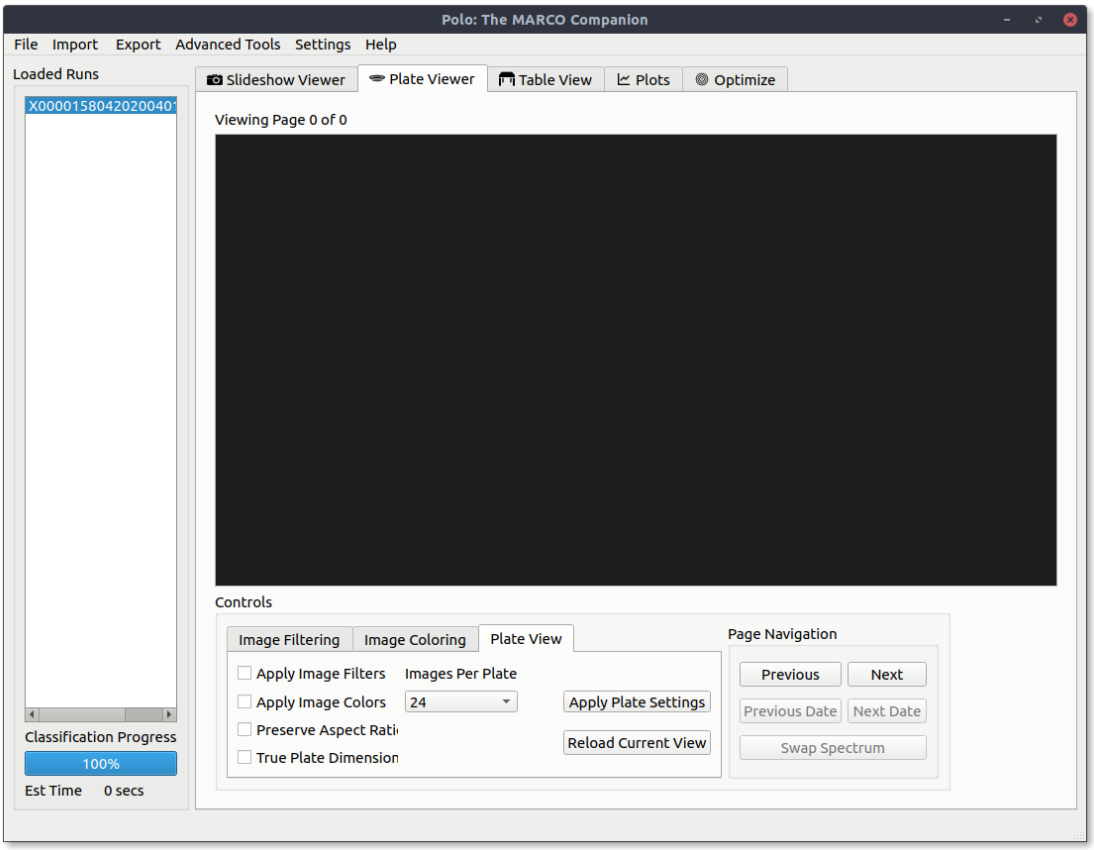
You can reset the slideshow to include all images by selecting all checkboxes or no boxes and pressing submit filters.

4.2.4 Image Metadata

Image metadata will be displayed in the **Image Details** and **Cocktail Details** windows when it is available. Image details will give you basic information about the image currently being displayed, such as well number, imaging technology imaging date and current classifications. If you are viewing an HWI run the chemical conditions the current image was plated in will be displayed in the **Cocktail Details** window.

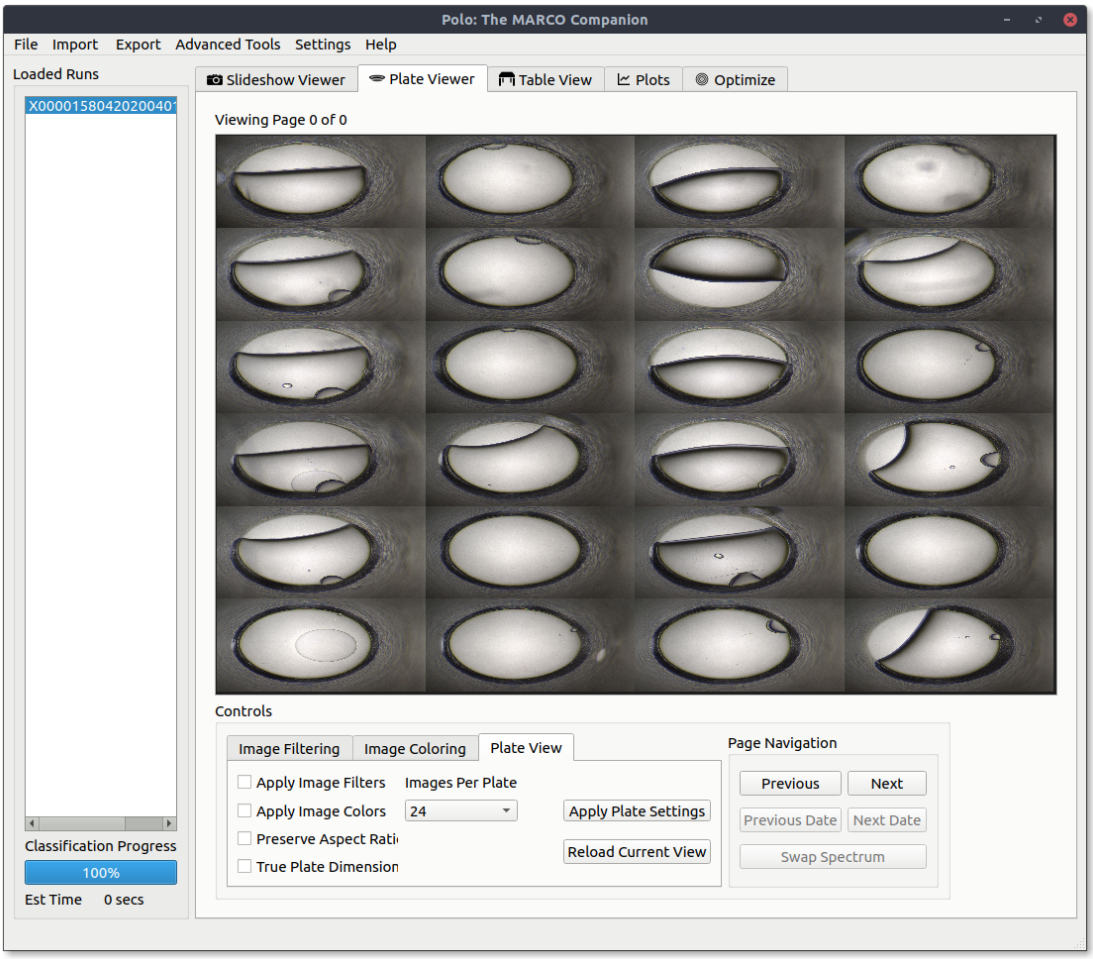
4.3 Using the Plate Viewer

To view multiple images in a grid you can utilize the **Plate Viewer**, which can be found under the **Plate Viewer** tab. When you first open it up, it will look something like the image below.

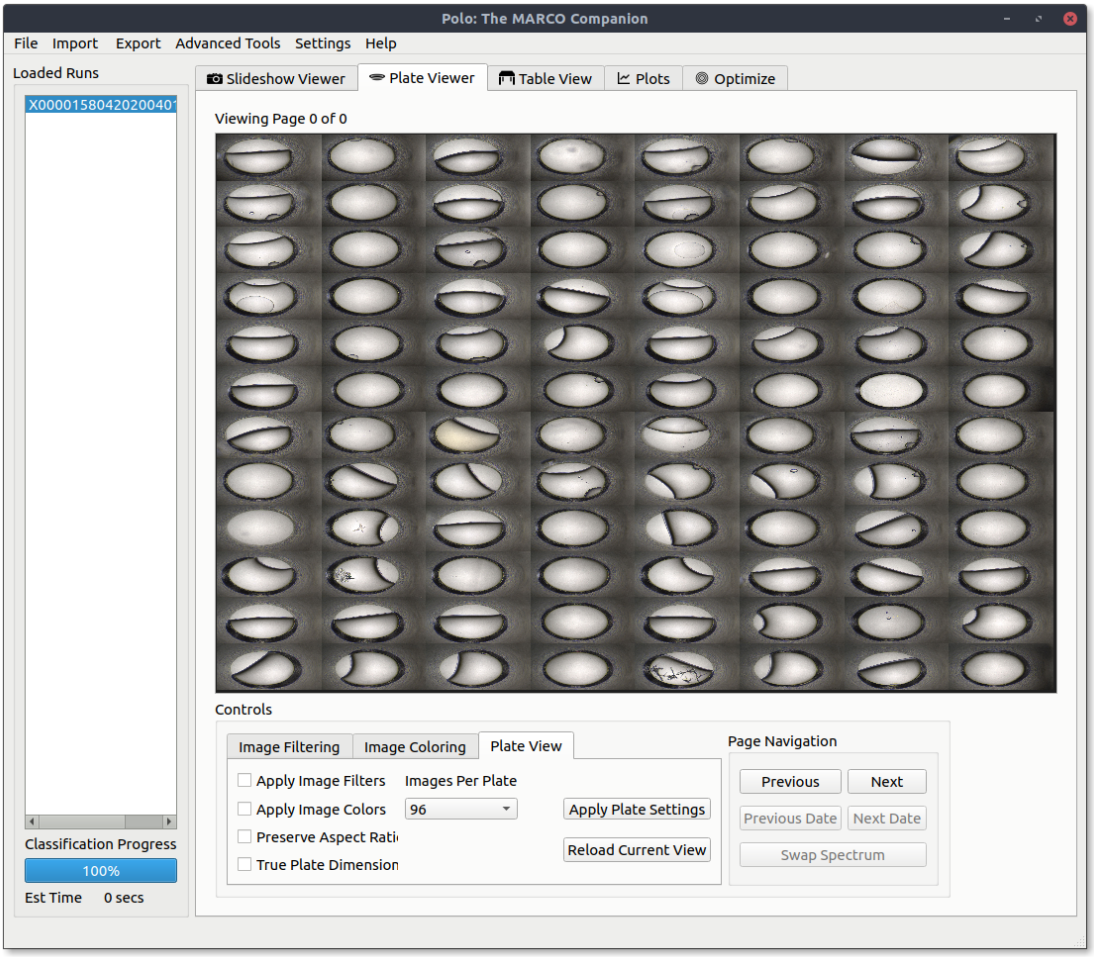


4.3.1 Basic Controls

Assuming you have a run loaded and selected press **Reload Current View** button to load in some images with the current settings.



You can adjust the number of images shown in the grid by using the **Images Per Plate** combo box and pressing **Reload Current View**.



You can navigate to the next or previous view by using the **Next** or **Previous** buttons respectively.

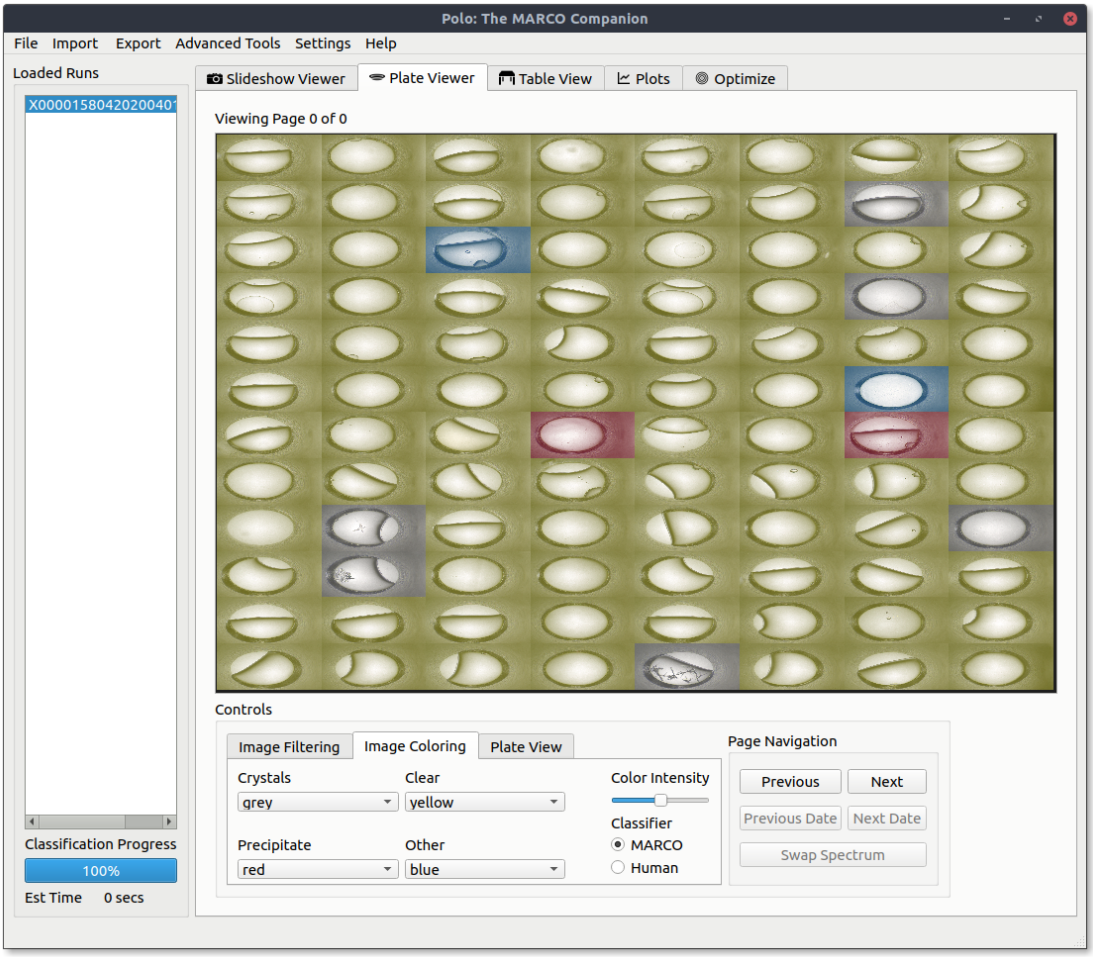
4.3.2 Image Filtering and Coloring

The Plateviewer window allows you to highlight images by either their MARCO classification or the one you have given them. This allows you to find true hits faster. First we will look at how to color images by their classification.

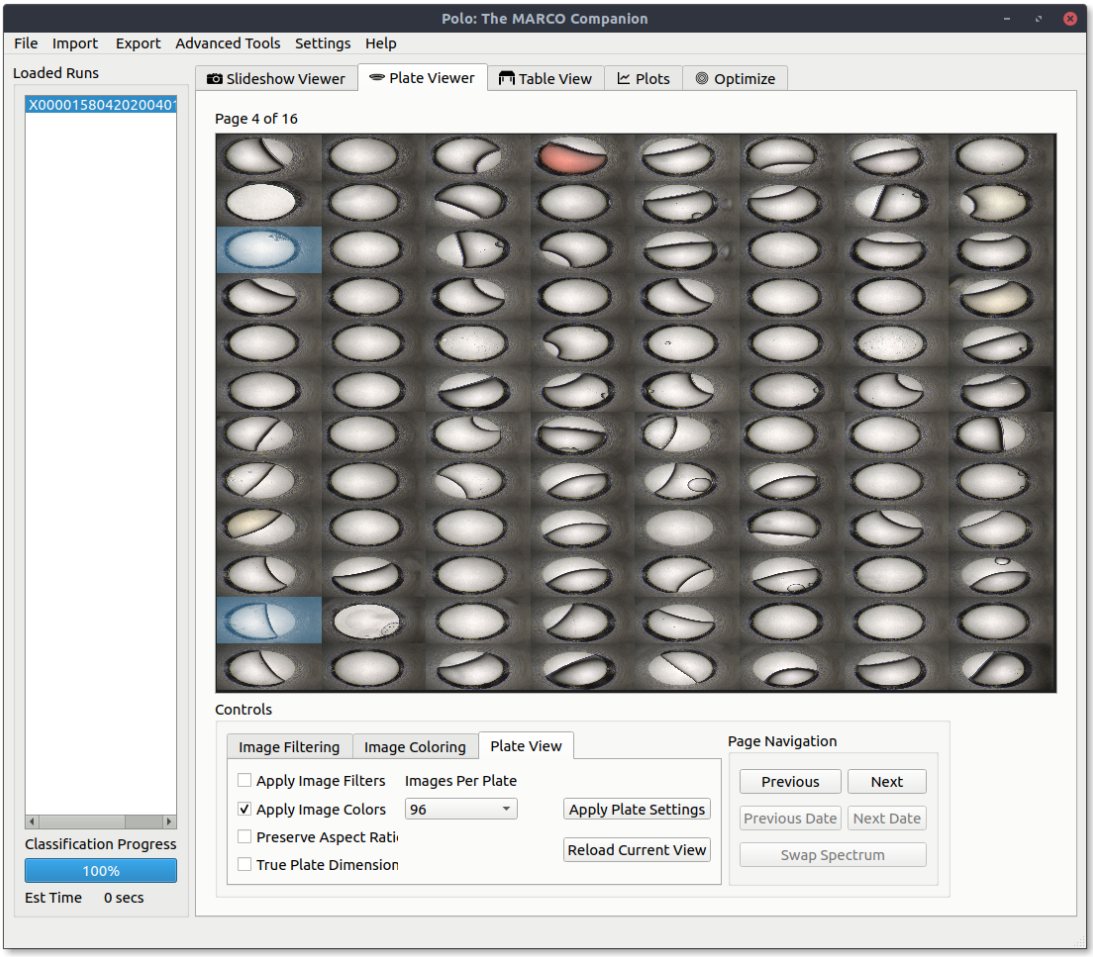
Open the **Image Coloring** tab on the bottom of the Plateviewer window. Use the combo boxes to assign a color to each classification type. The **MARCO** and **Human** radiobuttons tell Polo what classifier to use. After you have picked out a color scheme switch back to the **Plate View** tab select the **Apply Image Colors** checkbox and press the **Apply Plate Settings** button to color your images.

Note: The **Apply Image Colors** checkbox must be selected for colorings to be applied.

A 96 well view with colors applied to all MARCO classifications.

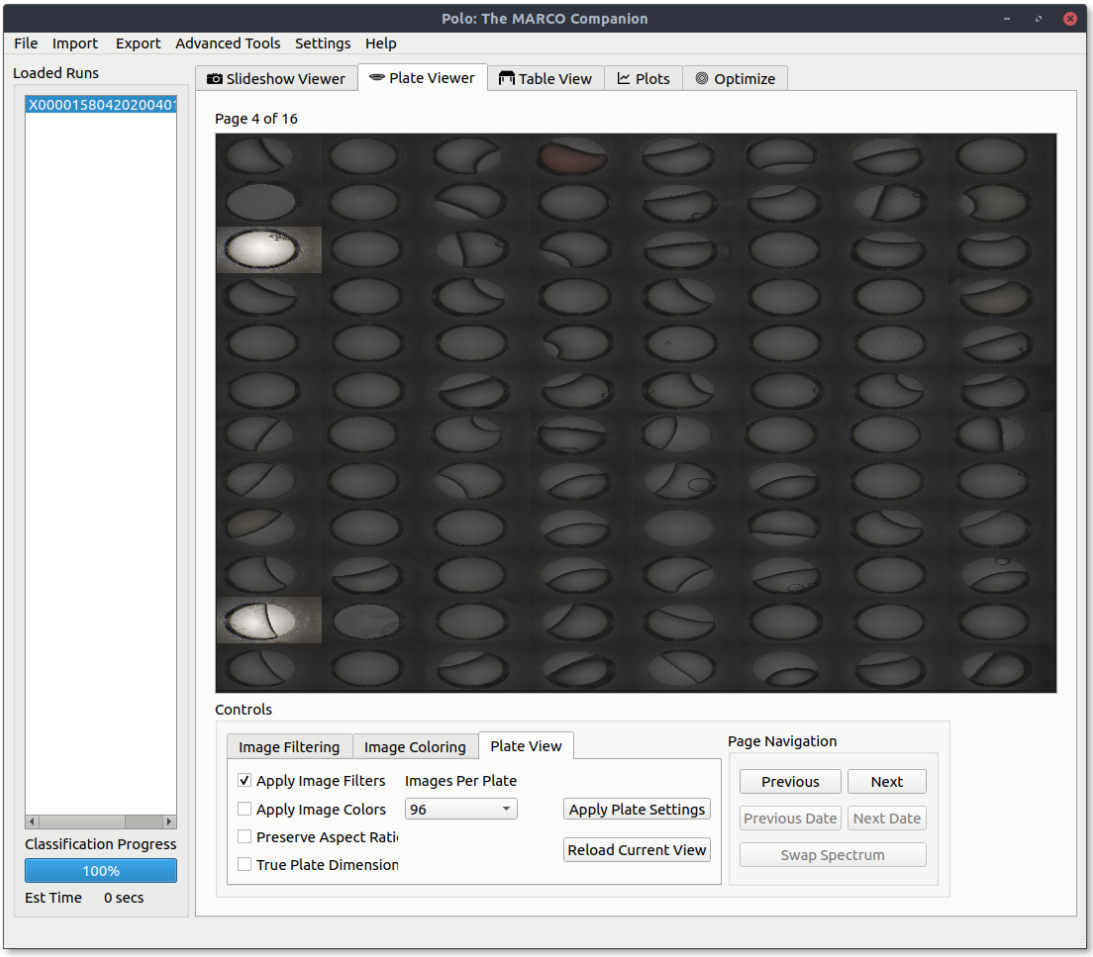


A disappointing 96 well view with only the Crystal classified images colored blue.



Similarly, images can also be emphasized and deemphasized by either their human or MARCO classification. Use the checkboxes under the **Image Filtering** tab to select which images to emphasize. Then switch back to the **Plate View** tab and check the **Apply Image Filters** box and hit **Apply Plate Settings**.

The same view as above but only selecting for MARCO classified crystal images.



4.3.3 Detail View

If you see an image that looks interesting you can select it and it will be opened in a popout window like the one below. Here you can view details about the image and assign it a classification using the buttons in the **Human Classification** panel.

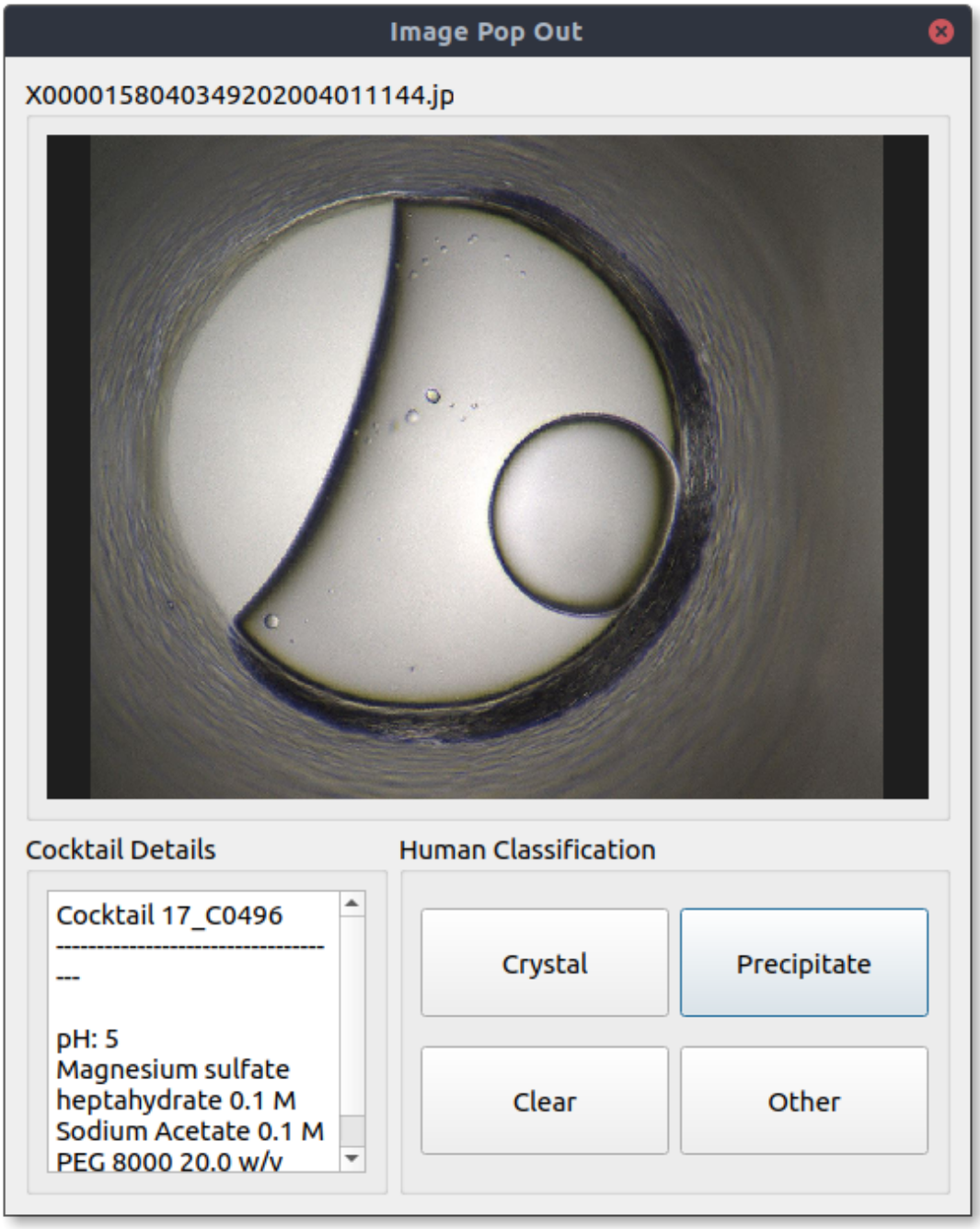
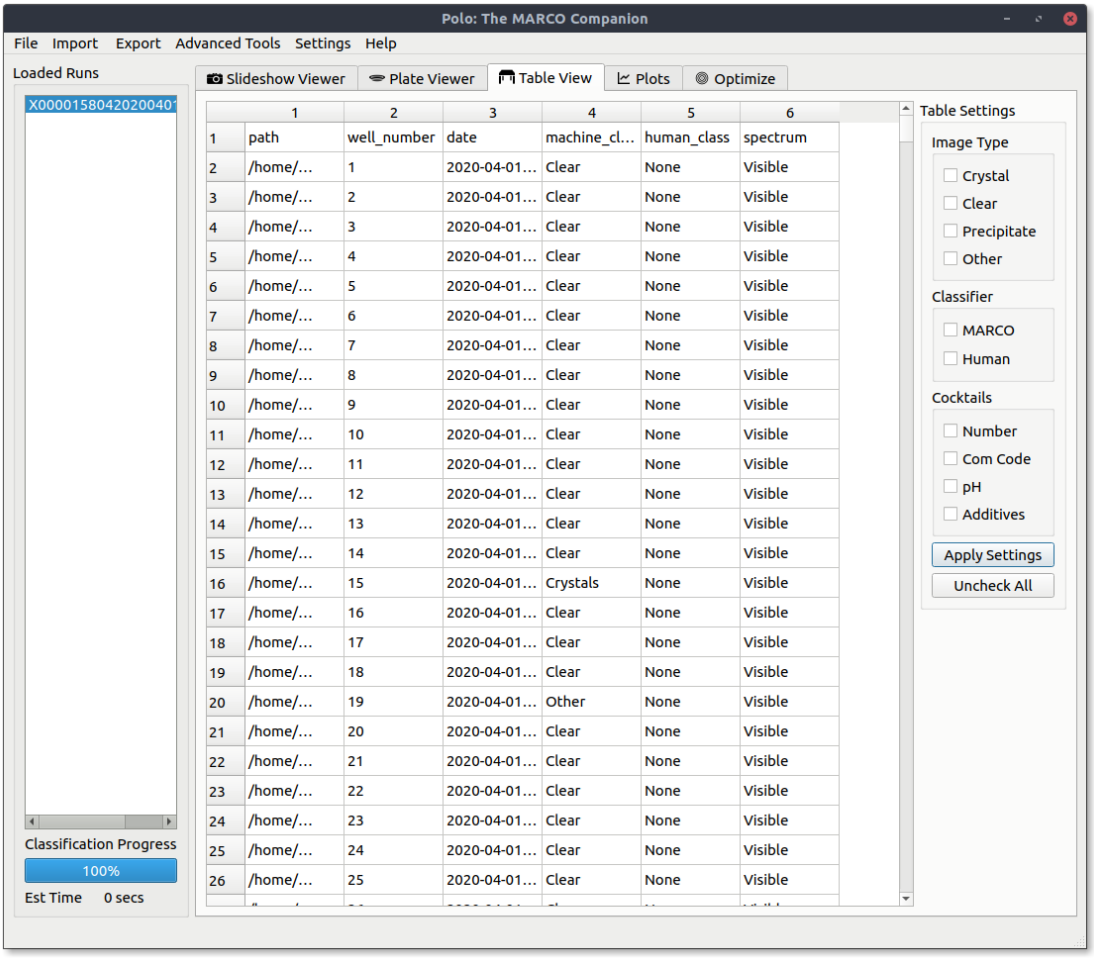


Image Pop Out
X00001580403492020040111144.jp
Cocktail Details
Cocktail 17_C0496
---
pH: 5
Magnesium sulfate heptahydrate 0.1 M
Sodium Acetate 0.1 M
PEG 8000 20.0 w/v
Human Classification
Crystal
Precipitate
Clear
Other

4.4 Using the Table View

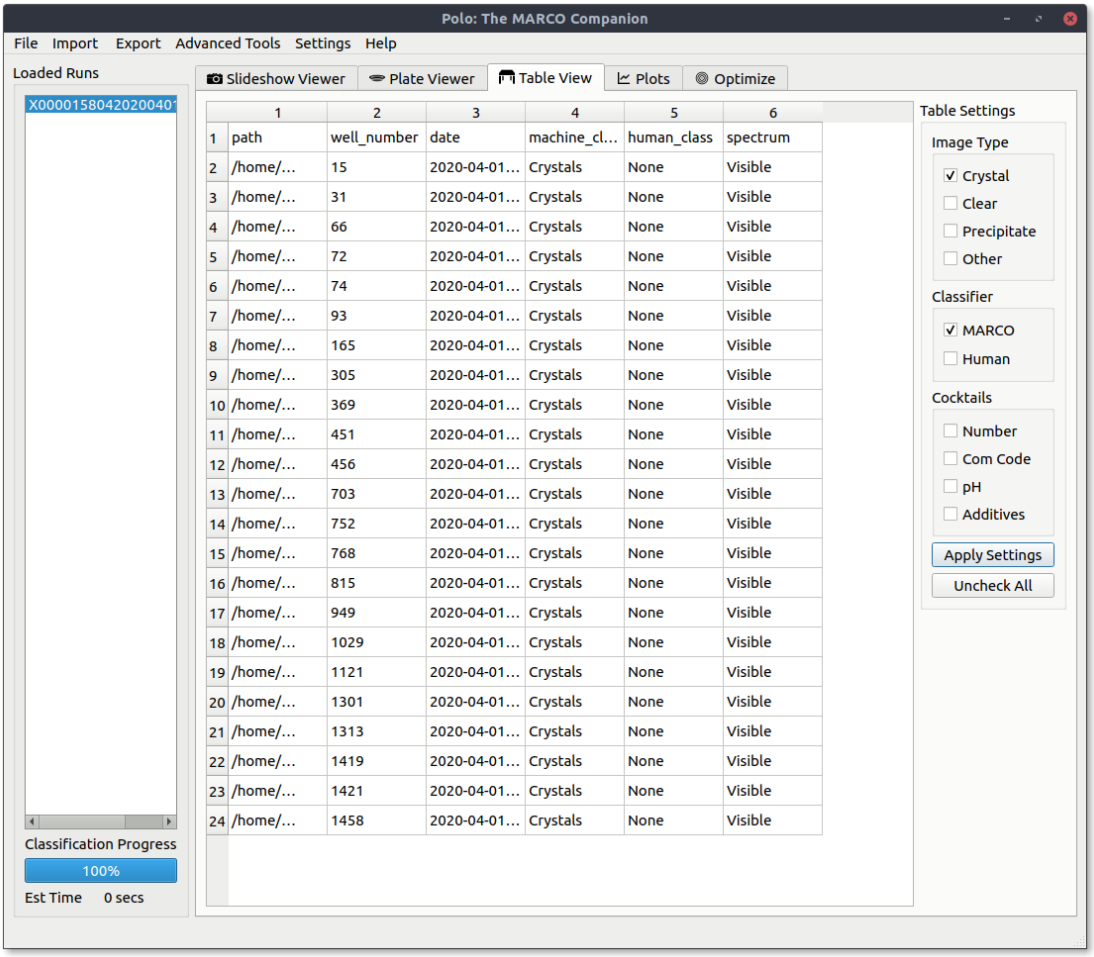
The table view allows you to view a run in a spreadsheet like view and does not display images.



	1	2	3	4	5	6
1	path	well_number	date	machine_cl...	human_class	spectrum
2	/home/...	1	2020-04-01...	Clear	None	Visible
3	/home/...	2	2020-04-01...	Clear	None	Visible
4	/home/...	3	2020-04-01...	Clear	None	Visible
5	/home/...	4	2020-04-01...	Clear	None	Visible
6	/home/...	5	2020-04-01...	Clear	None	Visible
7	/home/...	6	2020-04-01...	Clear	None	Visible
8	/home/...	7	2020-04-01...	Clear	None	Visible
9	/home/...	8	2020-04-01...	Clear	None	Visible
10	/home/...	9	2020-04-01...	Clear	None	Visible
11	/home/...	10	2020-04-01...	Clear	None	Visible
12	/home/...	11	2020-04-01...	Clear	None	Visible
13	/home/...	12	2020-04-01...	Clear	None	Visible
14	/home/...	13	2020-04-01...	Clear	None	Visible
15	/home/...	14	2020-04-01...	Clear	None	Visible
16	/home/...	15	2020-04-01...	Crystals	None	Visible
17	/home/...	16	2020-04-01...	Clear	None	Visible
18	/home/...	17	2020-04-01...	Clear	None	Visible
19	/home/...	18	2020-04-01...	Clear	None	Visible
20	/home/...	19	2020-04-01...	Other	None	Visible
21	/home/...	20	2020-04-01...	Clear	None	Visible
22	/home/...	21	2020-04-01...	Clear	None	Visible
23	/home/...	22	2020-04-01...	Clear	None	Visible
24	/home/...	23	2020-04-01...	Clear	None	Visible
25	/home/...	24	2020-04-01...	Clear	None	Visible
26	/home/...	25	2020-04-01...	Clear	None	Visible

4.4.1 Filtering

Just like the **Slideshow Viewer** and **Plate Viewer** tabs you can filter what data is shown to you in the table view. For those familiar with SQL this is like a SELECT / WHERE statement. Press the **Apply Settings** button to apply your currently selected filters.



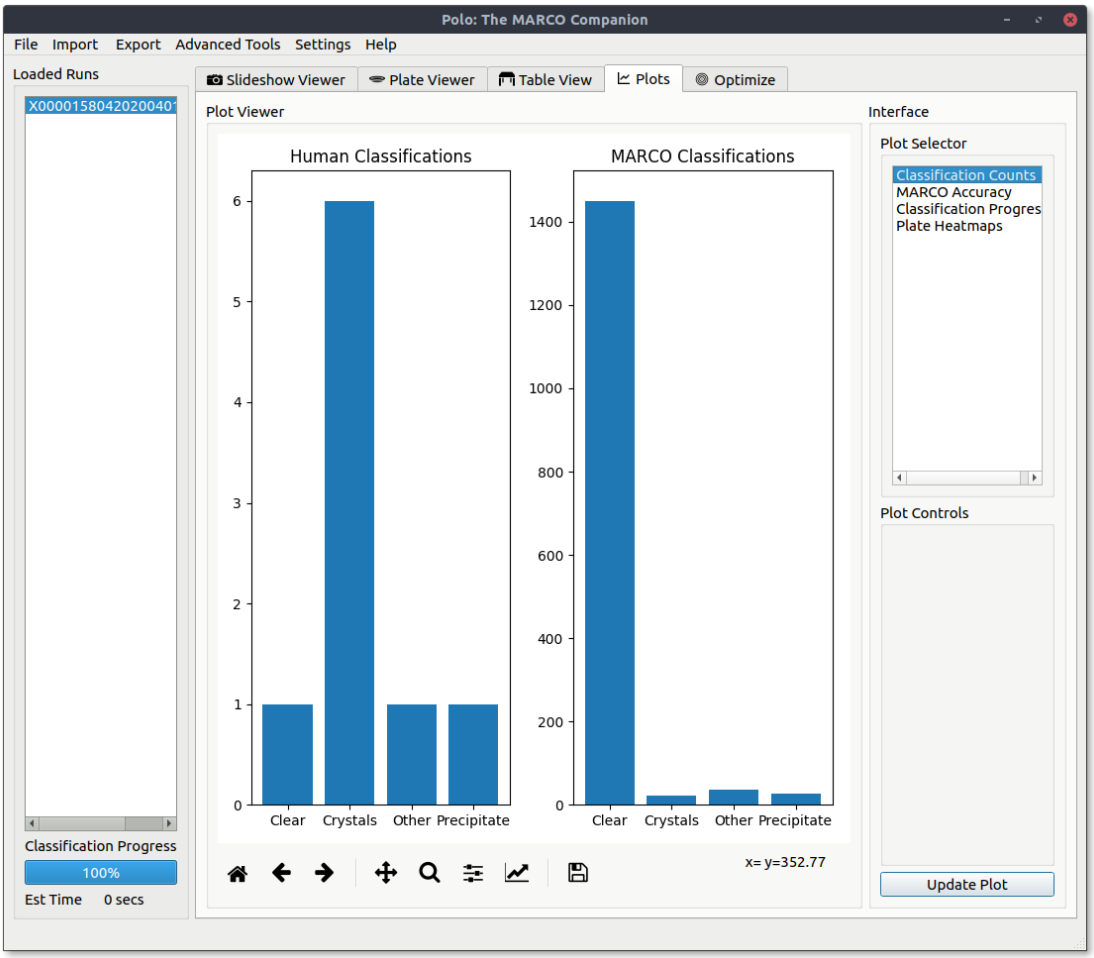
	1	2	3	4	5	6
1	path	well_number	date	machine_cl...	human_class	spectrum
2	/home/...	15	2020-04-01...	Crystals	None	Visible
3	/home/...	31	2020-04-01...	Crystals	None	Visible
4	/home/...	66	2020-04-01...	Crystals	None	Visible
5	/home/...	72	2020-04-01...	Crystals	None	Visible
6	/home/...	74	2020-04-01...	Crystals	None	Visible
7	/home/...	93	2020-04-01...	Crystals	None	Visible
8	/home/...	165	2020-04-01...	Crystals	None	Visible
9	/home/...	305	2020-04-01...	Crystals	None	Visible
10	/home/...	369	2020-04-01...	Crystals	None	Visible
11	/home/...	451	2020-04-01...	Crystals	None	Visible
12	/home/...	456	2020-04-01...	Crystals	None	Visible
13	/home/...	703	2020-04-01...	Crystals	None	Visible
14	/home/...	752	2020-04-01...	Crystals	None	Visible
15	/home/...	768	2020-04-01...	Crystals	None	Visible
16	/home/...	815	2020-04-01...	Crystals	None	Visible
17	/home/...	949	2020-04-01...	Crystals	None	Visible
18	/home/...	1029	2020-04-01...	Crystals	None	Visible
19	/home/...	1121	2020-04-01...	Crystals	None	Visible
20	/home/...	1301	2020-04-01...	Crystals	None	Visible
21	/home/...	1313	2020-04-01...	Crystals	None	Visible
22	/home/...	1419	2020-04-01...	Crystals	None	Visible
23	/home/...	1421	2020-04-01...	Crystals	None	Visible
24	/home/...	1458	2020-04-01...	Crystals	None	Visible

4.5 Using Plot Functions

Polo utilizes the Matplotlib python library to provide a few diagnostic plots to give you more information on your screening run. Once a run is loaded in plots can be viewed by selecting the **Plots** tab.

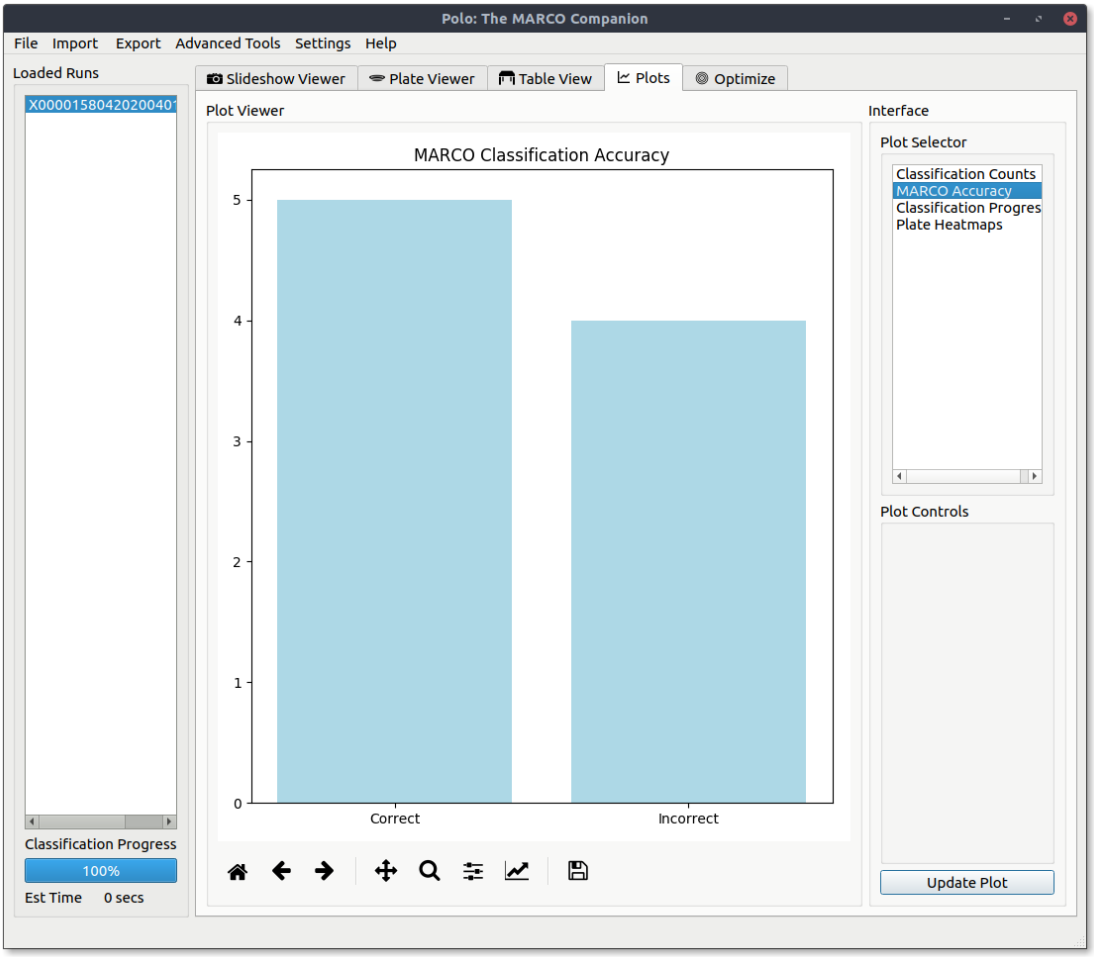
4.5.1 Classification Counts

Visualize human and MARCO classifications by image type.



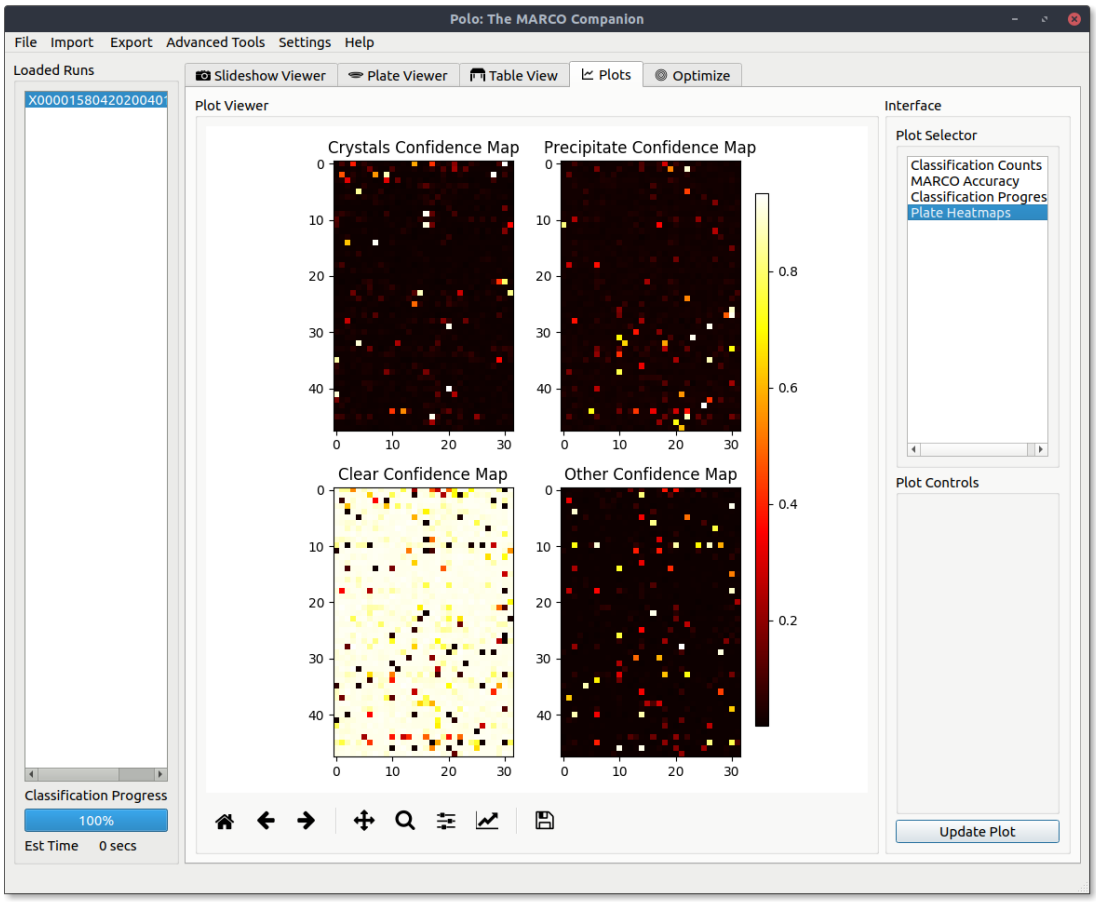
4.5.2 MARCO Accuracy

View a basic bar graph of MARCO classification accuracy. Correct classifications are those where the model applied the same classification as the human.



4.5.3 Plate Heatmaps

View the MARCO model confidence for each image classification across all images in your screening run.



4.5.4 Saving a Plot

If you wish, you can save a plot as an image to your machine by using the plot control icons at the bottom of the **Plots** tab. Select the floppy disc to save the current plot.

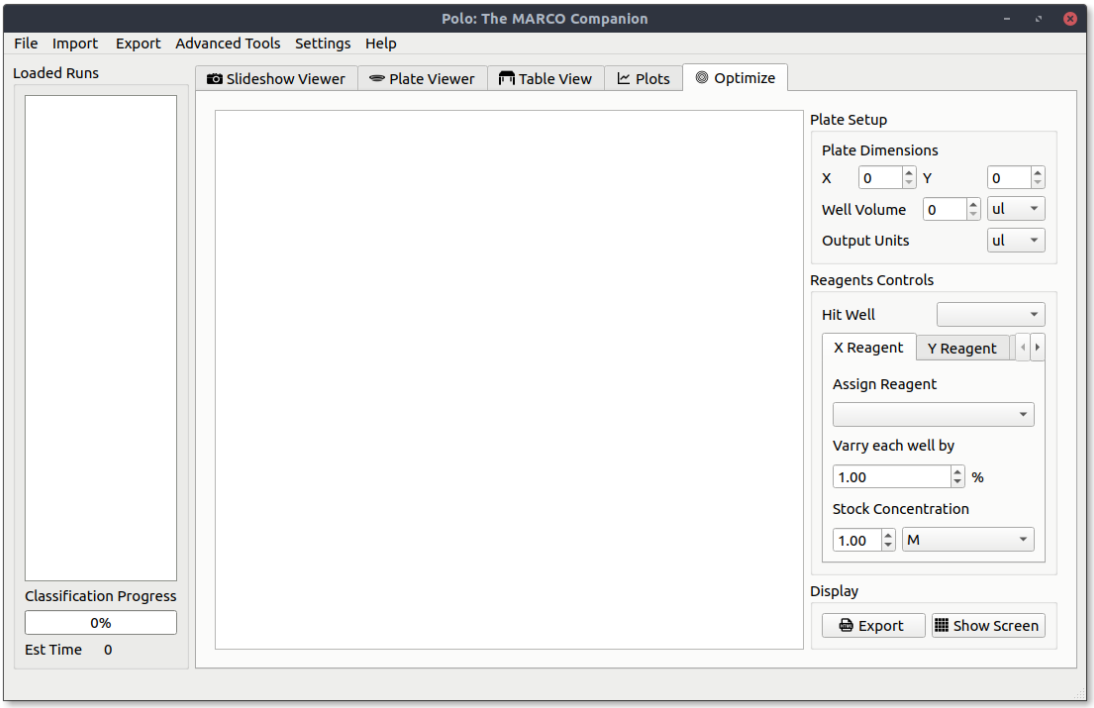
4.6 Using the Optimize Tool

Once you have identified crystal containing wells, you can easily design optimization screens using the optimize tool. The optimize tool automatically creates screens around the cocktail conditions a crystal hit grown in.

Currently all conditions come directly from HWI screening conditions and changing the conditions manually is not currently supported.

For more information on optimization protocols you can read this document provided by HWI,

Reproducing HWI HTCSC crystallization screening hits



4.6.1 Setting Up Your Plate

The first thing to do is to tell Polo what the plate you will be using for your optimization screen. Set the number of wells in your plate by adjusting the plate dimensions.

For example a standard 24 well plate could have 6 wells on X axis and 4 on the Y or vice versa. Which value you assign to which axis is arbitrary as long as you assign the reagent you want to screen for on that same axis.

You should also adjust the well volume by using the **Well Volume** and associated units combo boxes. This volume will be used as the maximum volume for each well.

4.6.2 Selecting Reagents

Once your plate is set up you are ready to select reagents to screen for. Wells you have classified as crystal containing are listed in the **Hit Well** combo box under the **Reagent Controls** panel. Selecting a well will display its containing reagents under the **Assign Reagents** combo boxes of the **X Reagent** and **Y Reagent** tabs.

Note: A reagent assigned under the **X Reagent tab** will be varied on the X-axis of the plate and the reagent assigned under the **Y Reagent** tab will be varied on the Y-axis of the plate.

Since any given plate only has two axis only two reagents can be screened for, but many crystallization cocktail contain three or more reagents. Reagents not assigned to either the x or y reagent will be considered constant and will be present in your final screen but always at the same concentration.

Once you have assigned your reagents set a stock concentration for each reagent and select a percentage to the x and y reagents by. This is always in reference to the hit concentration for the particular reagent.

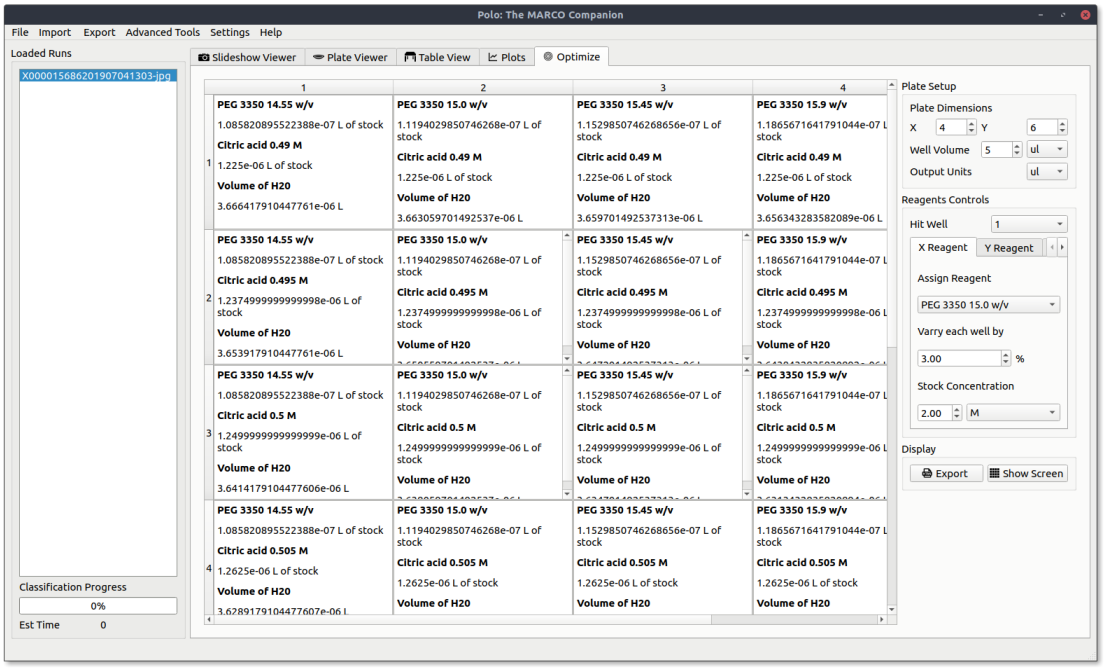
For example if you selected citric acid as your x reagent and it was present in the crystallization cocktail at a concentration of 0.5 M the concentration of citric acid in your optimization screen will be varied in reference to 0.5 M by the

percentage selected in the **Vary each well by** spin box.

Additionally, the hit concentration is always the center most well of the axis. Meaning if there are 6 wells on your plate's x-axis you can expect the concentration of citric acis in the 3rd well to be 0.5 M (using the example above).

4.6.3 Viewing Your Optimization Screen

After everything is set up, press **Show Screen** to display your current optimization screen. It will look something like the image below.



If it does not fill the screen slightly adjust the size of the Polo window and it should snap into place.

4.6.4 Exporting A Screen

After setting up your screen you can export it to an html file to print and / or share with your friends. To do so just hit the **Export** button. This will open a file browser and you can select where you would like to save the file. You can then open the file with any browser to view it or save it as a pdf.

4.7 Advanced Tools

Advanced tools go a bit beyond image classification and viewing and allow you to take advantage of the fact a single screening run will likely be imaged at multiple points in time and with multiple imaging technologies. All tools described below can be found by navigating to the menu bar and selecting **Advanced Tools**.

4.7.1 Time Resolved Runs

Usually, the sample plate will be imaged at multiple points in time. Polo allows you to link these runs together using the **Add Time Resolved** tool under **Advanced Tools**. Polo will automatically determine the order of the runs based on their date and then allow you to navigate between them in both the Slideshow Viewer tab and Plate Viewer tab using the **Next Date** and **Previous Date** buttons.

4.7.2 Adding an Alternative Spectrum

Oftentimes, the same plate will be imaged with a photographic technology besides microscopy in order to verify the presence of crystals in a sample. If you have images taken with such a technology you can identify them as such in the **Run Importer** window (link here).

Then navigate to **Advanced Tools** and select **Add Spectrum** which will open a dialog like the one shown below.

Valid runs will automatically be assigned to an image category based on the spectrum they were assigned at import. Select one run from each category that you would like to link and when finished press **Submit Assignment**.

You will then be able to swap between these images in both the Slideshow Viewer and Plate Viewers. This can be very useful for verifying a crystal is in fact a protein crystal and not salt.

4.8 Saving a Run

Once you have invested time into picking out the best screening conditions you are going to want to save your work and share it with others. The best way to do this in Polo is by saving your current run as a .xtal file.

4.8.1 .xtal File Format

.xtal is a json based format for saving screening runs after they have been loaded into and processed using Polo. It will maintain all of your image classifications and metadata in a single file. Additionally, the actual screening images are encoded directly into the xtal file which makes sharing your data to another Polo user very easy as it only requires sharing the single .xtal file.

4.8.2 Actually Saving Your Run

To save your Run from Polo, navigate to the menu bar and and select **File -> Save** or **Save As**. If you have not saved your current run to a xtal file before a file browser will automatically be opened and you can specify a location and name for your save file.

Note: Do not close Polo while a run is being saved!

After the save is complete Polo will notify you with a popup telling you it is safe to close the program and your save as completed. You can now share your xtal file with anyone else using Polo!

4.9 Exporting a Run

In addition to saving screening runs as xtal files, Polo supports exporting to formats which can be viewed outside of the Polo application. Currently available export options are described below.

4.9.1 HTML Reports

4.9.2 CSV Exports

CHAPTER

FIVE

BETA TESTERS GUIDE

5.1 Thank You!

Thank you for taking some time out of your day to help me test Polo. It means a lot. On this page you will find info on how to get started, where to find test data, what to do with it and how to report any bugs that you find.

5.2 Getting Polo Beta Version

Visit the Downloads page to see all current releases. Download the most current one for your operating system. You also might want to skim over the About page to get a better idea of what Polo's use cases are.

5.3 Test Data

Test images are available for download from the SourceForge page here. Ideally you should download one file from the *file* from the xtal_files folder and one *folder* from the test_images folder. The difference between the two formats is xtal is a file format used by Polo to store images, annotations, and classifications all in one place while the folder is more similar to what a new user would download directly from HWI's FTP server when their screening images are available.

Once you have the test data downloaded and Polo installed on your machine you are ready to start testing. The next section will outline a couple of basic tasks to get you started.

5.4 What Now?

Now that you have got some data you can start working with it. Here is a list of things to do with your data and links to sections of the documentation on how to do it.

- Open the xtal file: *Importing Images from a Directory*
- Open the folder of images and classify them: *Importing a Saved Run*
- Classify images using the Slideshow Viewer: *Using the Slideshow View*
- Classify images using the Plate Viewer: *Using the Plate Viewer*
- Create an optimization screen: *Using the Optimize Tool*
- Do something unexpected in an attempt to break the program (**Highly Recommended!**)

For more ideas and help getting started check out the *User's Guide*

5.5 I Found a Bug

Thats great! If you have some GitHub knowhow you can report it on the issues on the issues page of the Polo repository or fill out this google form Thanks for your help!

CHAPTER

SIX

POLO PACKAGE

6.1 Subpackages

6.1.1 polo.crystallography package

Submodules

polo.crystallography.broke module

polo.crystallography.cocktail module

polo.crystallography.image module

polo.crystallography.make_screen module

polo.crystallography.run module

Module contents

6.1.2 polo.marco package

Submodules

polo.marco.run_marco module

polo.marco.run_marco.**classify_image**(*tf_predictor*, *image_path*)
Given a tensorflow predictor (the MARCO model) and the path to an image, runs the model on that image. Returns a tuple where the first item is the classification with greatest confidence and the second is a dictionary where keys are image classification types and values are model confidence for that classification.

param: tf_predictor: Tensorflow predictor object. Should be MARCO model in ready to roll form. param: image_path: String. Path to an image that will be classified.

polo.marco.run_marco.**get_images**(*images_path*)
returns a list of all file paths contained in the given directory.

polo.marco.run_marco.**load_image**(*file_path*)

polo.marco.run_marco.**load_images**(*file_list*)
Loads images from a list of paths (should be from get_images) in format that can be read by the tensorflow package

Module contents

6.1.3 polo.utils package

Submodules

polo.utils.exceptions module

polo.utils.ftp_utils module

polo.utils.io_utils module

polo.utils.math_utils module

Module contents

6.1.4 polo.threads package

Submodules

polo.threads.thread module

class polo.threads.thread.**ClassificationThread**(*run_object*)
Bases: *polo.threads.thread.thread*

change_value

estimated_time

run(*self*)

class polo.threads.thread.**FTPDownloadThread**(*ftp_connection*, *file_paths*, *save_dir_path*)
Bases: *polo.threads.thread.thread*

file_downloaded

run(*self*)

class polo.threads.thread.**GraphThread**(*run_object*)
Bases: *polo.threads.thread.thread*

class polo.threads.thread.**PlateThread**(*run*, *image_types*, *marco*, *human*, *start_index*, *end_index*, *graphics_view_dims*)
Bases: *polo.threads.thread.thread*

place_image_in_scene(*scene*, *image*, *pixel_map*, *x*, *y*)

retrieve_images()

run(*self*)

tile_images_in_scene(*images*)

class polo.threads.thread.**QuickThread**(*job_func*, *parent=None*, ***kwargs*)
Bases: *polo.threads.thread.thread*

QuickThreads are very similar to thread objects except instead of writing code that would be executed by the *run* function directly, the function to be run on the thread is passed as an argument to *job_func* and any arguments that the passed function requires are passed as key word arguements. When the thread is started the key word arguments are passed to the job_func and the results of the call are stored in the *results* attribute.

```
my_func = lambda x, y: x + y
x, y = 40, 60
my_thread = QuickThread(job_func=my_func, x=x, y=y)
# set up the thread with my_func and the args we want to pass
my_thread.start()
# my_thread.result will = 100 (x + y)
```

Parameters

- **job_func** (*[type]*) – function to execute on the thread
- **parent** (*[type], optional*) – [description], defaults to None

run(*self*)

class polo.threads.thread.**SaveThread**(*main_window*, *run_to_save*, *output_path*)
Bases: *polo.threads.thread.thread*

run(*self*)

class polo.threads.thread.**thread**(*parent=None*)
Bases: PyQt5.QtCore.QThread

Very basic wrapper class around Qthread class. Should be inherited by a more specific class and then the *run* method can be overwritten do run the code that should be executed on the thread instance.

Parameters **parent** (*QWidget, optional*) – parent widget, defaults to None

run(*self*)

Module contents

6.1.5 polo.widgets package

Submodules

polo.widgets.file_browser module

polo.widgets.optimize_widget module

polo.widgets.plate_inspector_widget module

polo.widgets.plate_viewer module

polo.widgets.slideshow_inspector module

polo.widgets.slideshow_viewer module

polo.widgets.table_inspector module

polo.widgets.table_viewer module

Module contents

6.1.6 polo.windows package

Submodules

polo.windows.ftp_dialog module

polo.windows.image_pop_dialog module

polo.windows.log_dialog module

polo.windows.main_window module

polo.windows.run_importer_dialog module

polo.windows.secure_dave_dailog module

polo.windows.settings_dialog module

polo.windows.spectrum_dialog module

polo.windows.time_res_dialog module

Module contents

6.1.7 polo.designer package

Submodules

polo.designer.UI_FTP_Dialog module

polo.designer.UI_HTML_Dialog module

polo.designer.UI_buld_classifier module

polo.designer.UI_image_pop_dialog module

polo.designer.UI_log_dialog module

polo.designer.UI_main_window module

polo.designer.UI_make_tray module

polo.designer.UI_optimizeWidget module

polo.designer.UI_optimize_widget module

polo.designer.UI_plate_inspector_widget module

polo.designer.UI_plot_maker module

polo.designer.UI_run_exporter module

polo.designer.UI_run_importer module

polo.designer.UI_secure_save_dialog module

polo.designer.UI_settings module

polo.designer.UI_slideshow_inspector module

polo.designer.UI_spectrum_dialog module

polo.designer.UI_table_inspector module

polo.designer.UI_test module

polo.designer.UI_time_res_dialog module

polo.designer.UI_time_resolved_dialog module

polo.designer.UI_unit_converter module

polo.designer.Ui_make_tray module

Module contents

6.2 Module contents

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

polo.marco, 32
polo.marco.run_marco, 31
polo.threads, 33
polo.threads.thread, 32