How to Prepare a Quantius Job and Analyze the Results

This document contains instructions for how to prepare images for Quantius and how to analyze the results. We have provided a worked example of each using data that comes with the code.

**Installing the Code**

System Requirements

The quantiuspipeline package requires only a standard computer. The runtimes below are generated using a Mac computer with 16 GB RAM, 4 cores with 2.7 GHz.

The code is supported for Mac operating systems. The code has been tested on the following systems:

* macOS Sierra version 10.12.6
* macOS High Sierra version 10.13.3

The code requires MATLAB and MATLAB’s Mapping Toolbox. The code has been tested on the following versions of MATLAB:

* MATLAB\_2015b
* MATLAB\_2017b

Installation

1. Download the code from <https://bitbucket.org/arjunrajlaboratory/quantiuspipeline/overview>.
2. Add the path to wherever you have put the quantiuspipeline folder to MATLAB’s path:

|  |
| --- |
| >> addpath(genpath(‘\*\*insert path to code here\*\*’)) |

Naming Convention of the Code

The function “preprocess\_quantius\_job.m” contains a series of supporter functions that collect the images you want to submit to Quantius and prepare them (crop the images, adjust the contrast, etc.).

The function “postprocess\_quantius\_job.m” contains a series of supporter functions that organizes the results, plots them over the submitted images, converts them to the coordinates of an un-cropped or un-tiled image, and averages annotations.

Each of these supporter functions is called “controller\_(description of function)”. These controller functions iterate through the results (by individual annotation, by annotations on each slice, etc.) and perform a particular action. Those actions are functions called “algorithm\_(description of function)”. These algorithm functions contain the actual algorithm for formatting the results, uncropping/tiling the coordinates, averaging the results, etc. If you want to use different algorithms, those algorithm functions are the code to change. Note that the exact algorithm may need to be tweaked for optimal performance for each type of job submitted to Quantius.

**Preprocessing – How to Prepare Images for Submission**

Prepare the Images for Submission

An example set of images can be found in quantiuspipeline 🡪 example\_data 🡪 preprocessing. If the example is followed exactly, the output should resemble the data found in quantiuspipeline 🡪 example\_data 🡪 preprocessing\_result.txt.

The example should take ~10 minutes to run. Generally, the run time is however long it takes the user to set the desired image processing parameters, plus a few minutes for MATLAB to prepare the images.

1. Make sure all images are tifs. The files can contain z-stacks or slices.

*NOTE: You can set different settings for each file. This means that if a z-stack is saved as a single file, each slice in the stack will be preprocessed using the same settings. If a z-stack is saved as individual slices, each slice could be preprocessed with different settings.*

1. All images must start with the same word. The first word can be anything, and the file names after the first word can vary. Any images with a different naming convention will be excluded.
2. The images can come from multiple directories, so long as the naming convention is the same for all images.

Run the Preprocessing in MATLAB

1. All images and files used for a particular job will be stored in a single folder – called the job folder. If necessary, change MATLAB’s working directory to where you would like that folder to be saved.

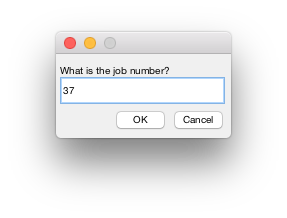
|  |
| --- |
| >> cd ~/Desktop |

1. Run the function “preprocess\_quantius\_job” to start the preprocessing. Follow the on-screen prompts to customize the job.

|  |
| --- |
| >> preprocess\_quantius\_job |

1. The first window will ask for a job number. Enter a number between 1 and 999.

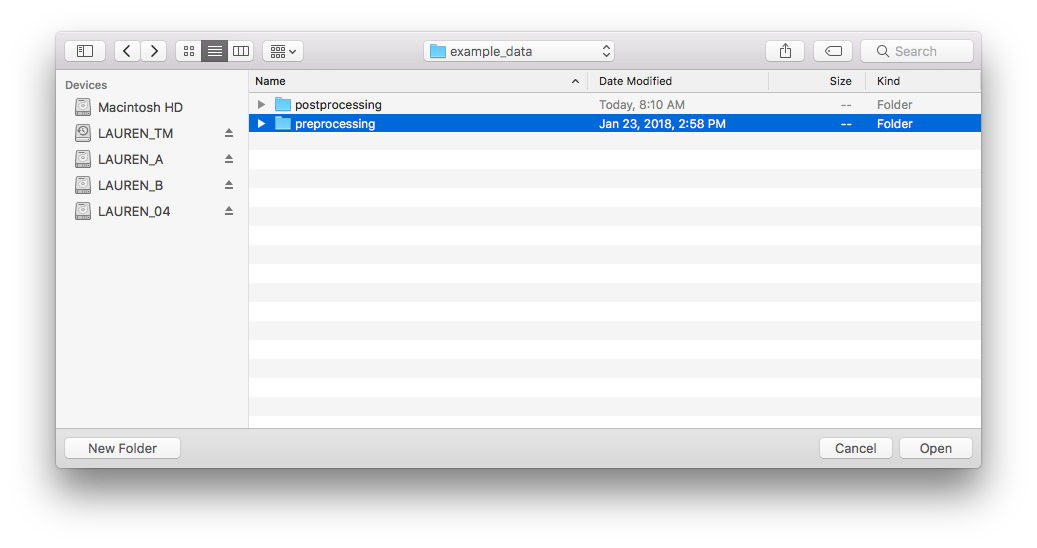
*NOTE: This assigns each Quantius job a unique number and saves all images and results with that number. This helps keep individual jobs separate, as well as ensure the images from a job are always associated with the results from the same job.*



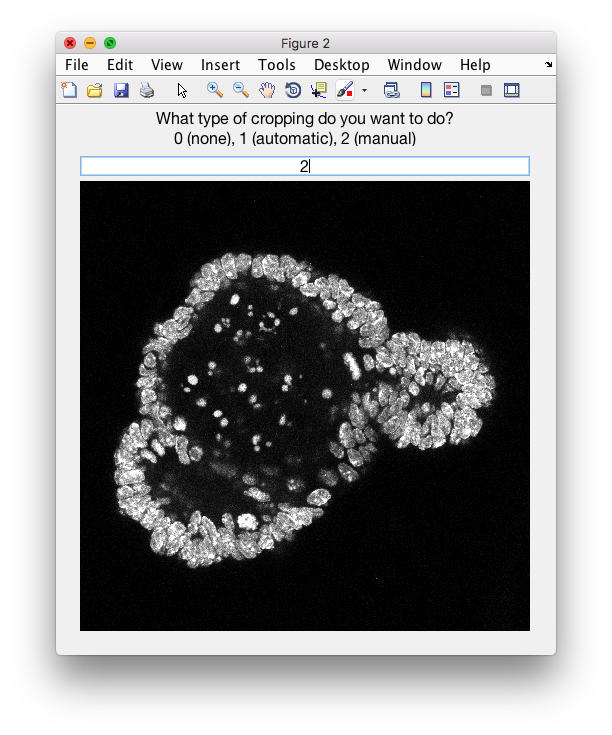
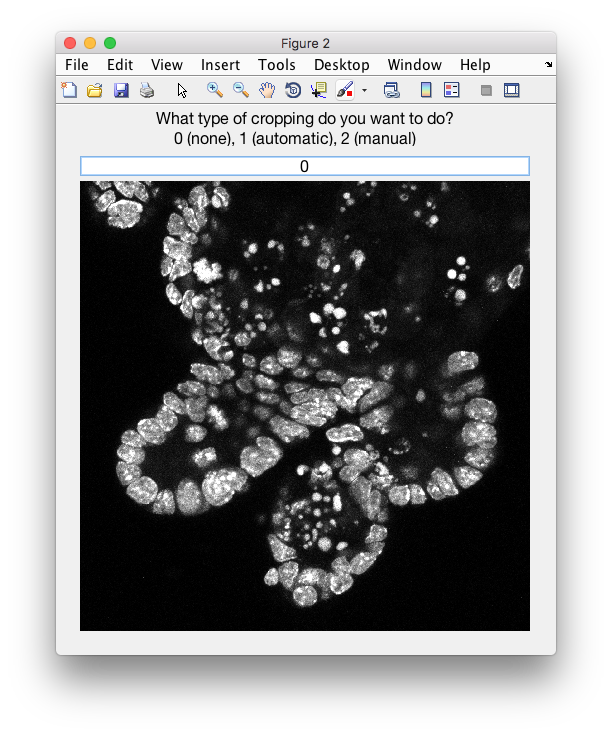
1. The next window will ask for the channel name. Enter the word that all files start with.

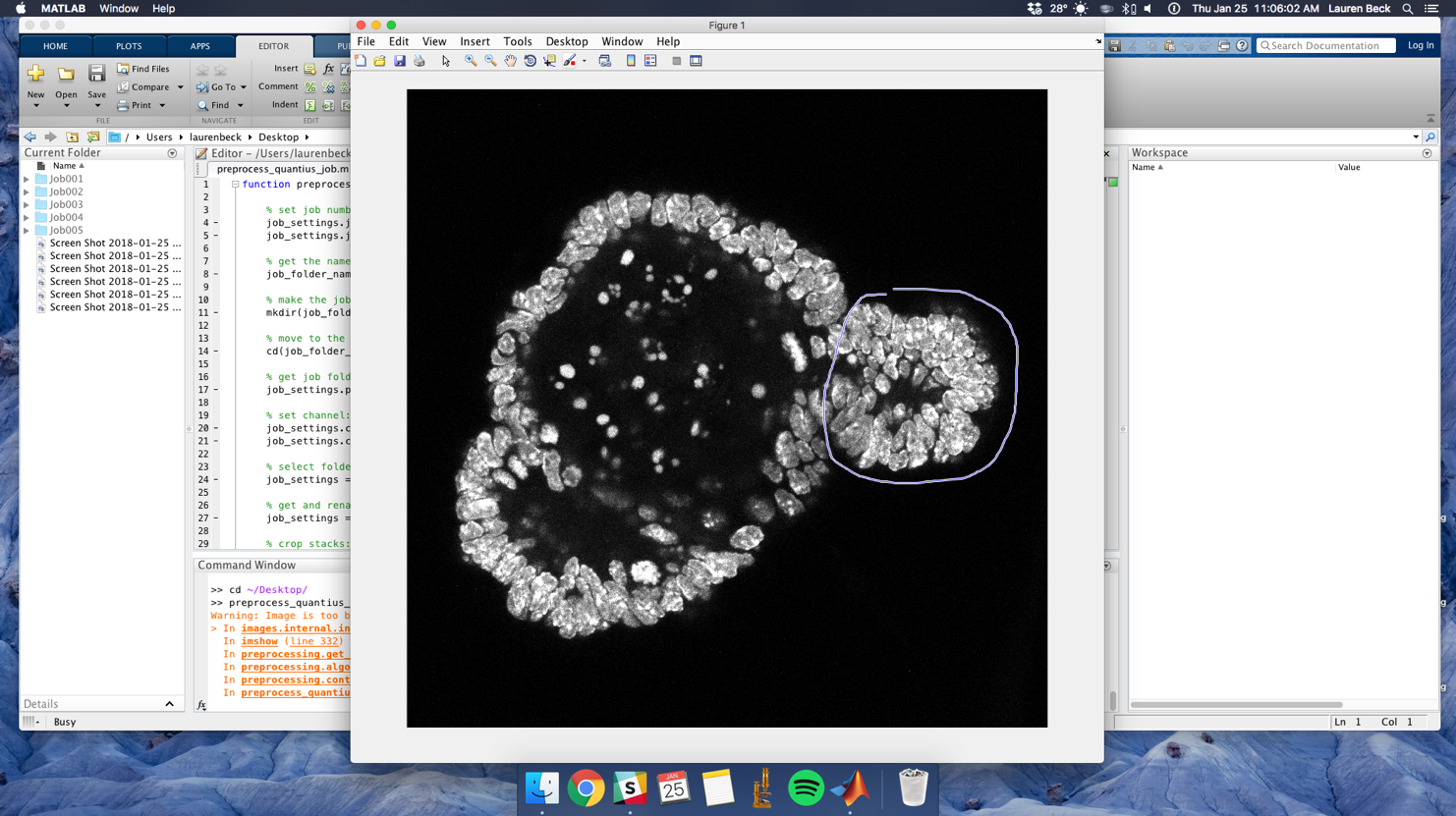


1. The next window will ask you to select the folder (or folders) containing the raw images.
   1. Navigate to the folder containing your raw images. Select that folder.
   2. Click “Open”.
   3. The window will pop open again. Repeat until you have opened all folders with raw images for the job.
   4. When you have selected the last folder, click “Cancel” the next time the window opens.



1. The next window asks what type of cropping you want to use, for each file.
   1. Determine the type of cropping to use.
      1. The window will display the image (if the file is a stack the image is the middle slice) with enhanced contrast.
      2. Enter what type of cropping you want to use on that file, 0 (no cropping), 1 (automatic cropping – use only when the image contains a single object), or 2 (manual cropping).
      3. Close the window when you have entered the value.
   2. Crop the image.
      1. If you entered 1, the image will be cropped automatically.
      2. If you entered 2, you will be prompted to draw a single region on the image. The region should be within the bounds of the original image.
   3. Repeat for each file.

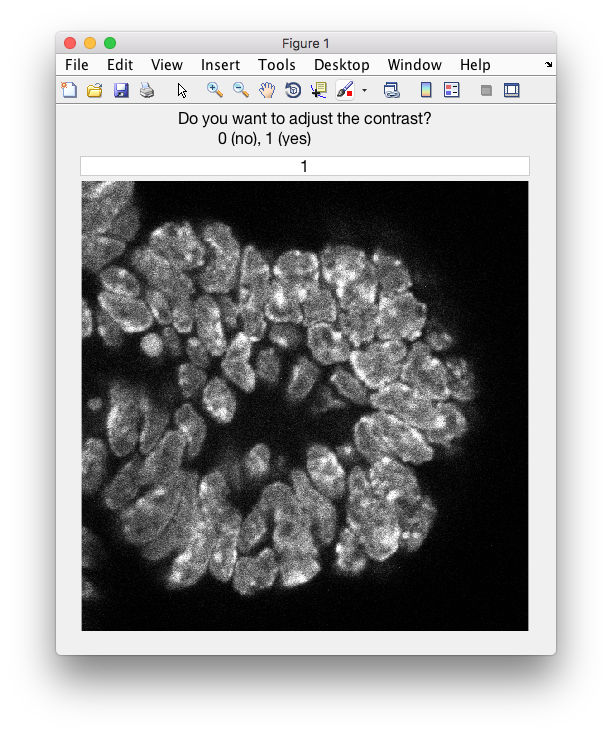
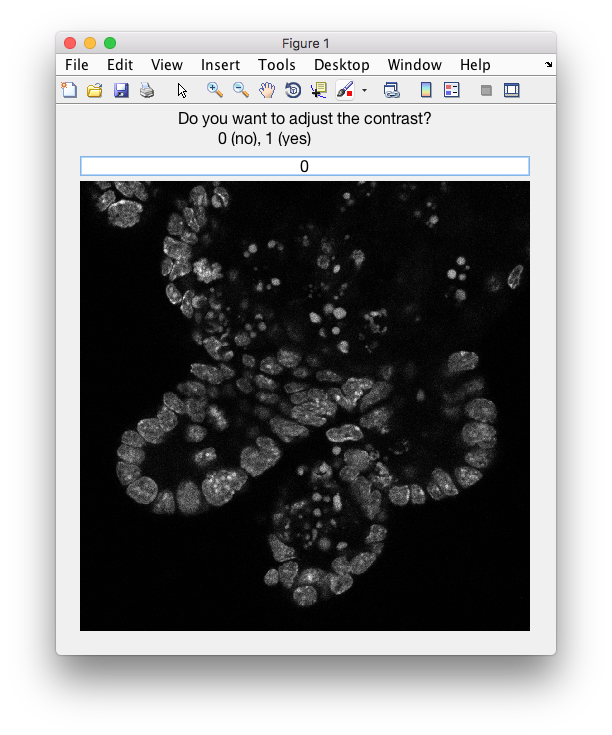


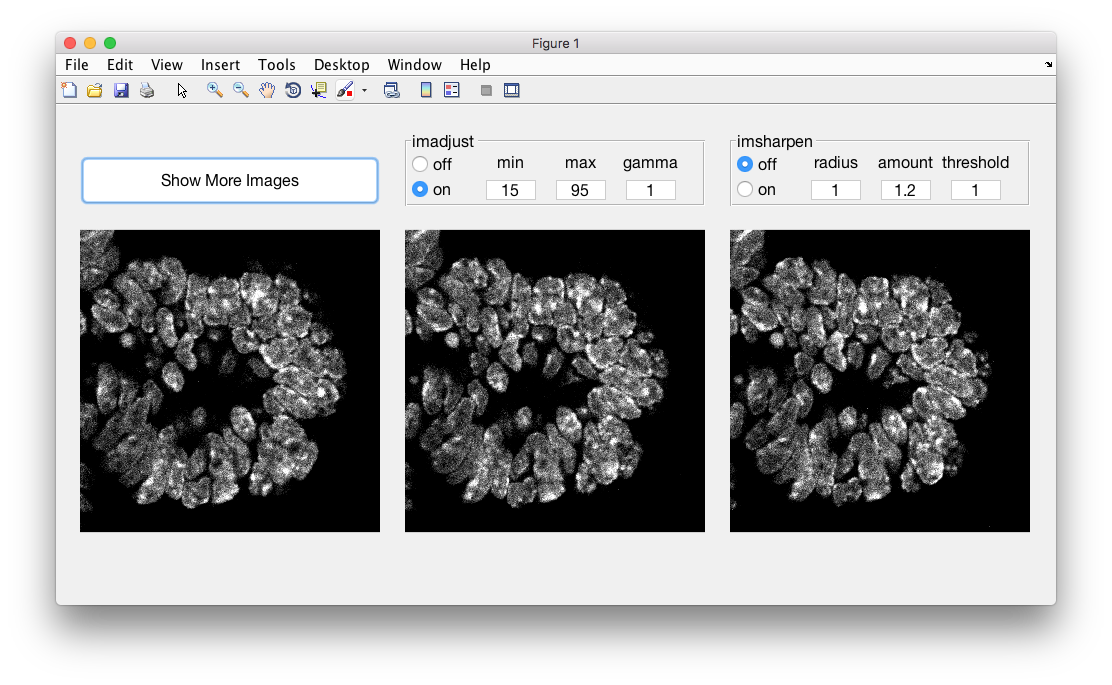


1. The next window asks what type of contrast adjustment you want to use, for each file.
   1. Determine whether or not to adjust the contrast.
      1. The window will display the image (if the file is a stack the image is the middle slice) with NO enhanced contrast.
      2. Enter what type of contrast adjustment you want to use on that file, 0 (do not adjust contrast) or 1 (adjust contrast).

*NOTE: Currently, the contrast adjustment ONLY works for files that were z-stacks.*

* + 1. Close the window when you have entered the value.
  1. Adjust the contrast.
     1. If you entered 1, you will be prompted to enter values for MATLAB’s [imsharpen](https://www.mathworks.com/help/images/ref/imsharpen.html?searchHighlight=imsharpen&s_tid=doc_srchtitle) and [imadjust](https://www.mathworks.com/help/images/ref/imadjust.html). The window will show the contrast adjusted images as the numbers are varied. The images displayed will be 3 random slices from the z-stack. Click “Show More Images” to see another 3 random images.
     2. Close the window when you have entered the values.
  2. Repeat for each file.





1. The next window asks what type of tiling you want to use, for each file.
2. Determine whether or not to tile the image.
   * 1. The window will display the image (if the file is a stack the image is the middle slice) with enhanced contrast.
     2. Enter what type of tiling you want to use on that file, 0 (do not tile) or 1 (tile).

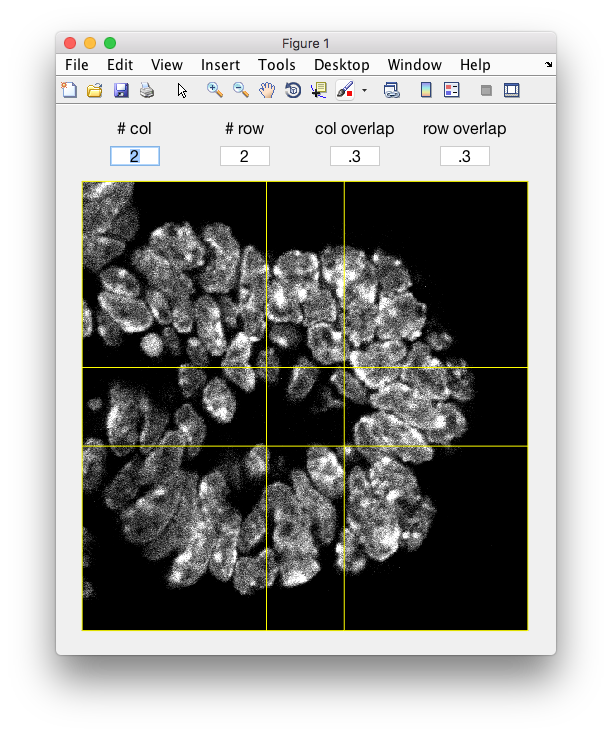
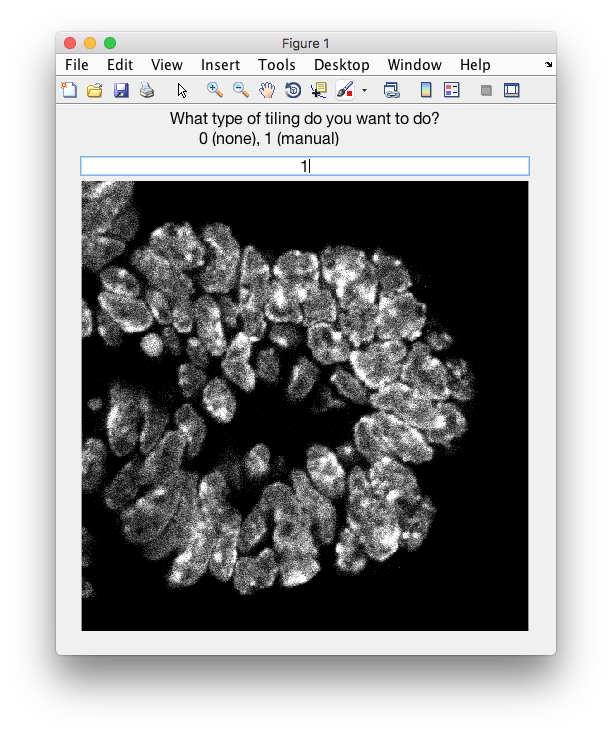
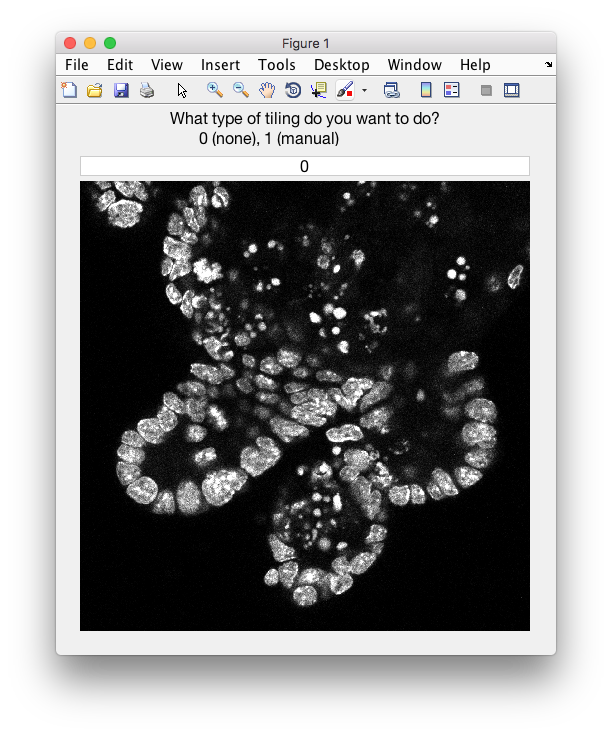
*NOTE: Tiling is helpful for getting quality annotations when there are a lot of objects in an image. Generally, the annotations are worse when there are more than 20 (for shape tools) - 50 (other tools) objects in an image.*

* + 1. Close the window when you have entered the value.

1. Tile the image (if necessary).
2. Enter the number of tiles to use per row and per column in the left two boxes. The values for each must be 1 or 2.
3. If the number of tiles for the image is > 1, then enter the overlap in the right two boxes. The overlap must be below 1 (though it can be 0). The overlap represents the fraction of the tile that overlaps with another tile.

*NOTE: Having overlap avoids an object being split between two tiles.*

1. Close the window when you have entered the values.
2. Repeat for each file.



1. The next window asks if you want to submit every slice, for each file.
   1. Determine whether you want to submit each slice of the z-stack to Quantius.
      1. Select “Yes” to submit all slices.
      2. Select “No” to submit every nth slice to Quantius (you will set n later).
   2. Set the frequency of slices to submit (if necessary).
      1. Enter a value for n. If you want to submit every 10th slice, enter 10 and slices 1, 11, 21, etc will be submitted.
   3. Repeat for each file.
      1. *NOTE: Since the images have already been tiled, each tile is now a separate z-stack.*



1. Submit the images in the folder “06\_submitted” to Quantius.

**Postprocessing – How to Analyze Results from Quantius**

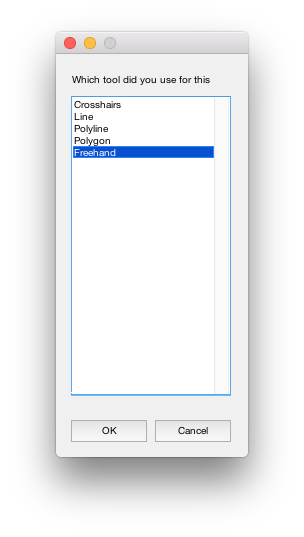
An example set of images and results can be found in quantiuspipeline 🡪 example\_data 🡪 postprocessing. If the example is followed exactly, the output should resemble the data found in quantiuspipeline 🡪 example\_data 🡪 postprocessing\_result.txt.

The example should take ~40 minutes to run. Generally, the run time scales with the number of annotations in each image, and the number of images submitted to Quantius. The run time will also be faster for point or line annotations (using the Quantius tools Crosshairs, Line, or Polyline) than for shapes (using the Quantius tools Polygon and Freehand).

1. Prepare to run the postprocessing:
   1. If following the example:
      1. Make the folder quantiuspipeline 🡪 example\_data 🡪 postprocessing you working directory in MATLAB.
   2. If analyzing your own data:
      1. Go to quanti.us and navigate to the job.
      2. Select “Download Data” then “JSON”.
      3. A new tab will open with the results. Right click on the page and select “Save As”.
      4. Save the file as “results\_1\_raw.json” in the folder with all the preprocessing images.
      5. Make that folder your working directory in MATLAB.
2. Run the function “postprocess\_quantius\_job” to do the postprocessing.

|  |
| --- |
| >> postprocess\_quantius\_job |

1. A window will ask which tool was used for the job. Select “Freehand”.



1. If the annotations on Quantius are shapes (i.e. the tool used was either the Freehand or Polygon tool), you will be asked if you want to connect the shapes in 3D.



How Postprocessing Works

|  |  |
| --- | --- |
| Step Description | Output |
| For all tools… | |
| 1. Load the “job\_settings.mat” file that contains all the preprocessing settings used for the job. |  |
| 1. Asks the user which tool was used to annotate the images. |  |
| 1. Formats the results into a structure where each annotation is an entry. | results\_2\_formatted.mat |
| 1. If necessary, removes shape annotations that touch the border of the submitted image. | results\_3\_formatted\_cleaned.mat |
| 1. Plots all raw results on the submitted images. All annotations are plotted on the image slice. They are color coded by turker. | 08\_results\_all folder |
| 1. Plots individual raw results on the submitted images. A folder is made for each turker and contains their annotations only. | 09\_results\_individual folder |
| 1. Converts the results back into original coordinates (from cropped or tiled coordinates). | results\_4\_converted.mat |
| 1. Clusters the annotations so you know, within an image, which annotations describe the same object. | objects\_1\_clustered.mat |
| 1. Plots the clusters on the original images. The annotations are color coded according to which cluster they belong to. | 10\_clusters folder |
| 1. Averages the members of each cluster to produce an average annotation for each image. | objects\_2\_averaged.mat |
| 1. Plots the average annotations on the original images. The annotations are color coded according to the cluster they came from. | 11\_averages folder |
| For polygon and freehand tools only… | |
| 1. Cleans the averages to eliminate situations in which there are two averages per object. | objects\_3\_averages\_cleaned.mat |
| 1. Plots the average annotations (in orange) that survived the cleaning and the annotations that did not survive the cleaning (in yellow). | 12\_averages\_clean folder |
| 1. Connects the objects across slices to create 3D segmentations. | objects\_4\_connected.mat |
| 1. Plots the annotations on the slices. The annotations are color coded by which 3D object they belong to. | 13\_connections\_2D\_slices folder |
| 1. Plots the annotations as slices in 3D. The annotations are color coded by which 3D object they belong to. | 14\_connections\_3D\_slices folder |
| 1. Plots the annotations as volumes in 3D. The annotations are color coded by which 3D object they belong to. | 15\_connections\_3D\_volumes folder |