# Aro Spot Counting Suite Tutorial

**For Dr. Paaby's Lab**

Diptodip Deb

A guide for and explanation of the Aro Spot Counting Suite
for use in counting cells in FISH images of worm embryos.

Paaby Lab, Biology Department
Georgia Institute of Technology

# Contents

# 1 Introduction

This is a guide to using the Aro Suite to count cells in worm embryos. Aro was created by Scott Rifkin <sarifkin@ucsd.edu>. Aro uses a machine learning approach called the Random Forest algorithm (Breiman, 2001) to classify bright spots in FISH microscopy images as either cells or not cells (spots or not spots). This allows for highly accurate ($\geq$95%) automated counts of thousands of cells across multiple worm embryos.

# 2 Preparation

In order for Aro to work, you have to do a bit of prep work. First you have to set up MATLAB to be able to use Aro, then you have to make sure your images are named and organized in a certain way, and then you have to make mask files for your images or image stacks. If you have already set up Aro in MATLAB, then skip to Section 2.3. If not, then follow the instructions below.

## 2.1 A note about directories for Aro

Aro is particular about the way that files and folders and named and set up.

First, you need to set up a folder for your current project. This could be something like `DataSet` for example. This folder will be referred to as the working directory or the top directory.

Within the working directory, there should be a folder called `ImageData` that will contain the image `.stk` files. Within `ImageData`, there should be a folder called `SegmentationMasks` that contains the image mask `.tiff` files. The rest of the folders will be set up later and semi-automatically by Aro Suite. For a list of all the folders you should have at the end, see Figure 1.

## 2.2 Setting up Aro in MATLAB

You'll need to set up Aro in MATLAB if you want to run it locally (as opposed to on the PACE/other compute cluster).

1. Download the Aro Suite from
   <https://github.com/diptodip/AroSpotFindingSuite/archive/master.zip>
   if you have not already done so.
2. Extract the zip file and copy the resulting folder to `~/Documents/MATLAB`.

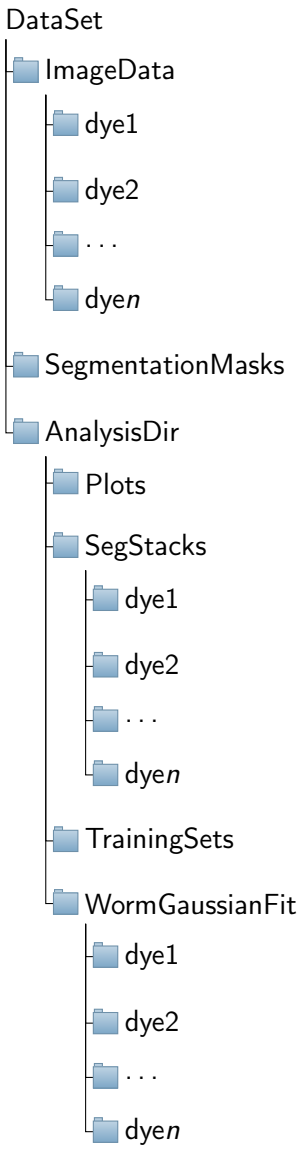


Figure 1: A folder tree showing the directory setup for using Aro.

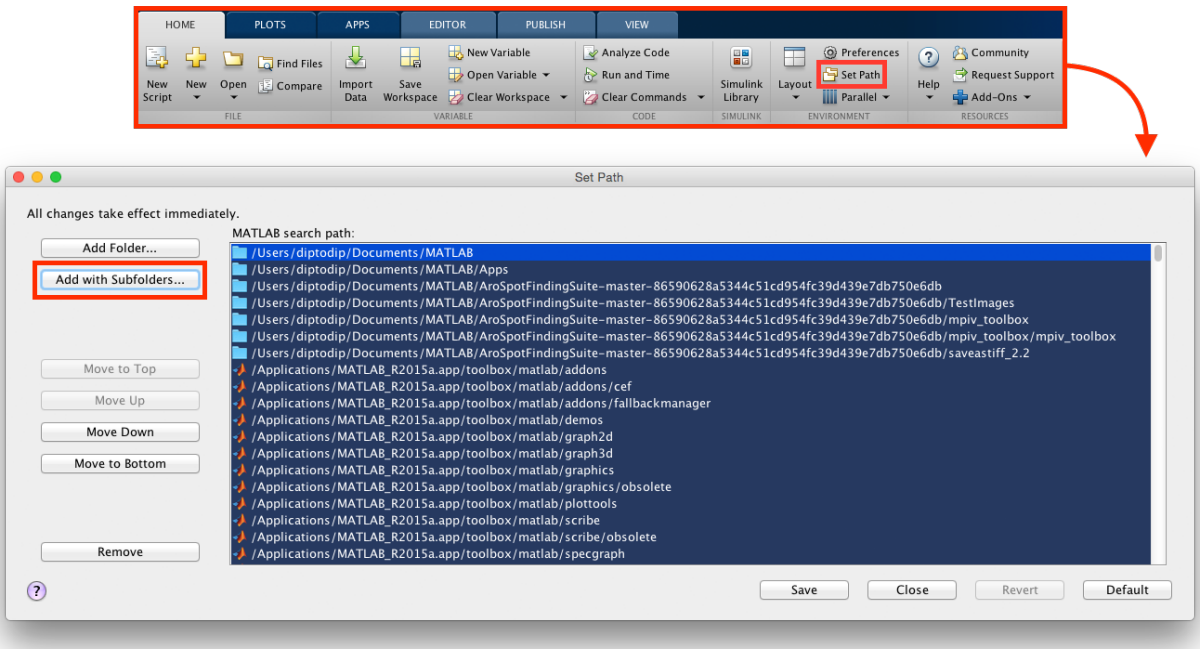


Figure 2: Screenshots that show how to set up the Aro files in MATLAB.

3. Open MATLAB and on the `HOME` tab, find the `ENVIRONMENT` section. Click on `Set Path` (see Figure 2).

4. In the dialogue box that pops up, click on `Add With Subfolders`.

5. Navigate to `~/Documents` and select the `~/MATLAB` folder that has the Aro Suite folder inside. Then hit `Open`.

6. Hit Save. Ignore any warnings about saving `pathdef.m` and needing administrator rights and just hit OK to all of them.

## 2.3 Prepare your image files

Aro is also very picky about the way that your image files are named and organized — but it has to be. This is how it knows which files are which and

when to use which file. If you mess up your file naming, you could throw off the whole set of counts. Be careful!

1. In your working directory, make a folder called `ImageData` if you haven't already.

2. For each image file, place the file in the `ImageData` folder with the filename formatted as `dye_xxx.stk` where `dye` is the name of the `dye` and `xxx` is the 3 digit position number (e.g. position 1 is 001). Possible image formats include `.stk` and `.tiff` files.

### 2.4 Prepare masks for your images

In order for Aro to be able to identify distinct embryos you must provide mask images. You will make a separate mask image for each embryo in an image stack file. Each mask image will become a position later. This step will become more clear after following the instructions once through.

Note that we can automatically produce mask images with the command `auto_trim_mask_directory`. This command will both automatically trim the stacks in your image directory to the frames that show the embryo and produce a mask image for the embryo in the appropriate location. You can review these masks with the command `show_mask_preview` as long as you are in the top level directory (i.e. the directory containing the `SegmentationMasks` folder).

Howeever, we can also make masks manually. In order to make the mask images, we will use ImageJ. For macOS, you can find instructions on how to install ImageJ on their website at <http://imagej.nih.gov/ij/docs/install/osx.html>.

1. Open your stack file in ImageJ.

2. Select one of the embryos using the "Freehand selections" tool.

   - See Figure 4.
   - Make sure you are in the middle of the stack.
   - Make sure you do this with a steady hand.

3. In the menu, go to `Edit > Selection > Create Mask`.

4. In the menu, go to `Edit > Invert`.

5. Save the mask as a `.tiff` file by going to `File > Save As > Tiff...`

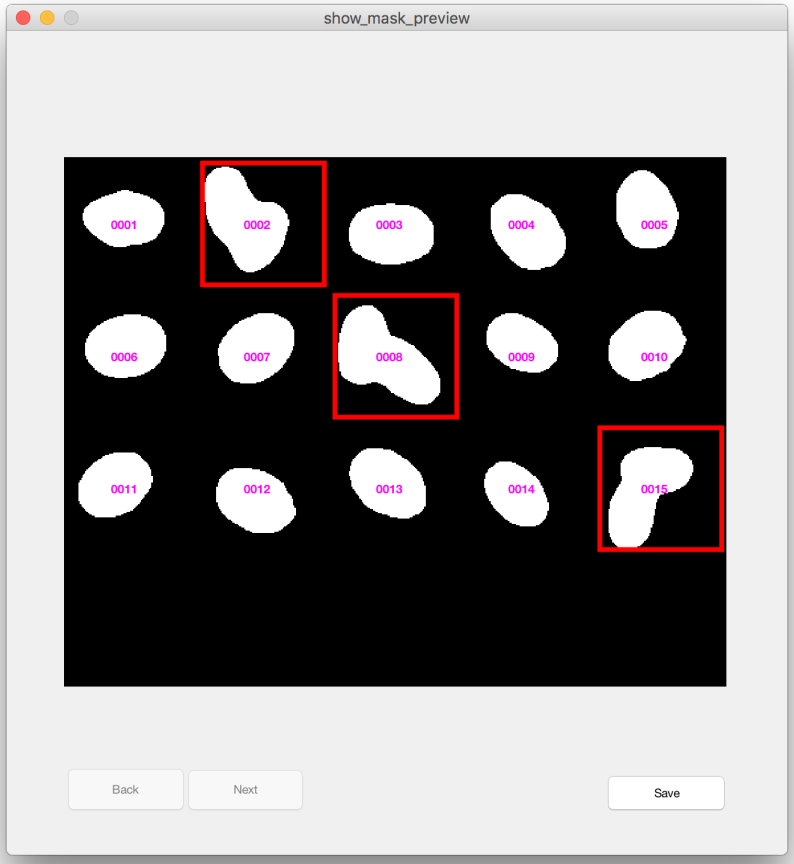


Figure 3: Screenshot showing the GUI for previewing the mask files. The red boxes indicate masks that have been determined to be bad masks. Aro will automatically run a check for predicting bad masks, but this will not find all bad masks/find false positive bad masks. Clicking a thumbnail will toggle the status of the mask file. Files with bad masks will not be analyzed.

- Make sure you name the file in the format: `Mask_xxx_y.tiff` where `xxx` is the 3 digit position number and `y` is the single digit mask number, e.g. `Mask_001_1.tiff` is the first mask for the first image (so it is the mask for the first embryo in the position 001 stack). All the mask files should be placed under the `ImageData/SegmentationMasks` folder.

### 2.5 Set up a parameter file for your data set

Now that your image folders are set up, you need to create a file that tells Aro where the files that it will look for are located. Keep in mind the folder structure shown in Figure 1.

1. In your working directory, place a copy of the `example-parameters.m` file that came with your copy of Aro.
2. Rename that file to `Aro_parameters.m`.
3. In MATLAB, open up this new parameters file.
4. Make sure that the `nestedOrFlatDirectoryStructure` option is set to equal `'nested'`.
5. Set the `dyesUsed` variable to be a cell array of strings representing the names of your dyes, i.e. `dyesUsed={'dye1',..., 'dyen'};` and e.g. `dyesUsed={'cy','tmr'};`.
6. Set the `AnalysisDir` variable to equal a string representing the name of the Analysis folder, e.g. `Analysis_ddMonthyy`.
7. Keep everything else the same.

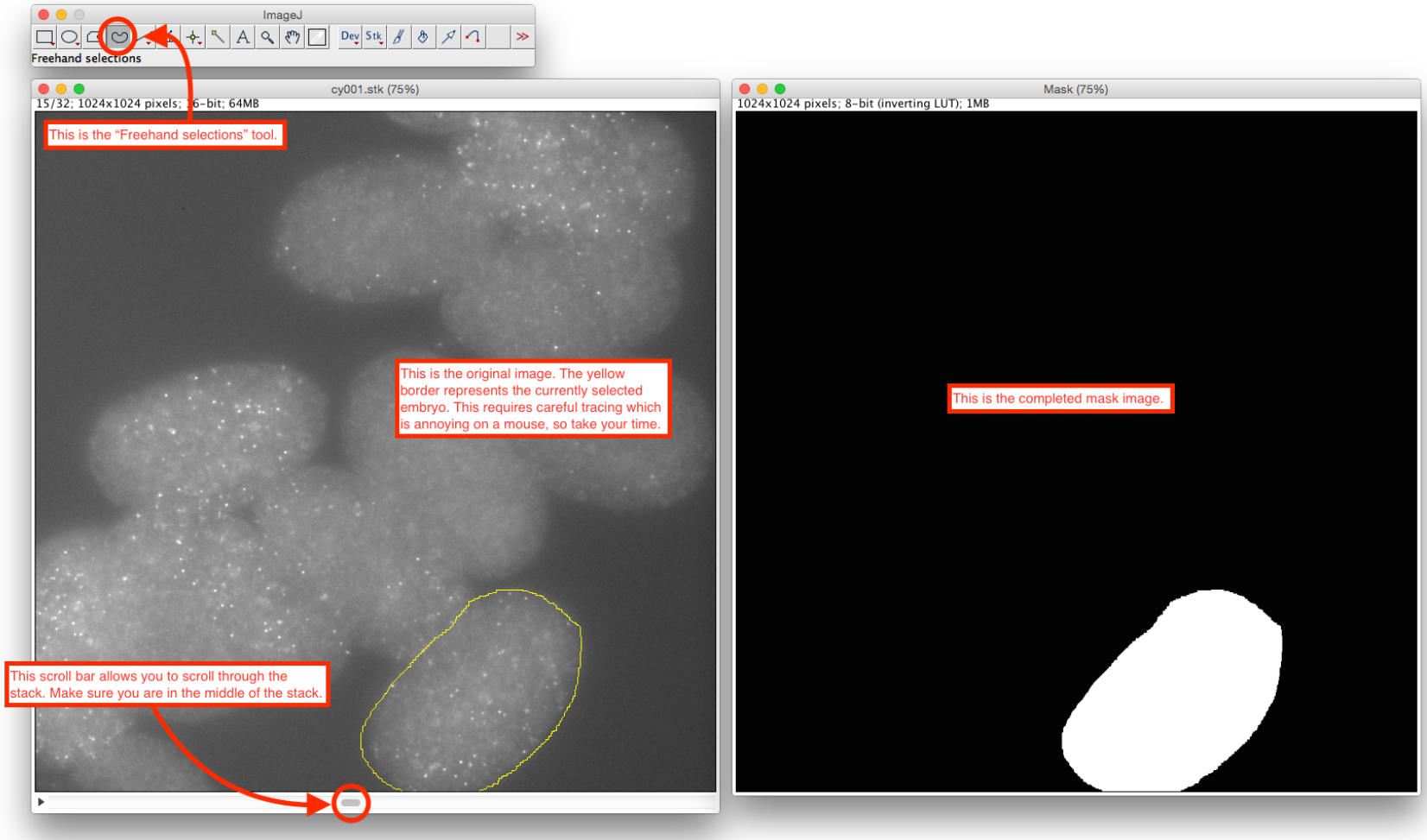


Figure 4: Screenshots showing the creation of a mask from a FISH image in ImageJ.

### 2.6 Create segmentation masks for Aro

After creating the mask files for all the images of embryos at all the positions, you need to run an Aro command that allows Aro to understand the mask files. This is a distinct step from the previous step in which we made masks for the images. Note that you **do not** need to do this if you have used the command `auto_trim_mask_directory` to produce the masks.

From the top directory, open MATLAB and in the command window run `createSegmenttrans('Posx')` if you want to process just the mask file for position x or `createSegImages('tif')` if you want to process all the mask files in the `SegmentationMasks` directory.

Each command will create a `dye_xxx_segStacks.mat` file that tells Aro where the embryos are for each dye and each position. By this point, the naming scheme should be obvious.

## 3 Classifying Spots With Aro

Now Aro should be ready to go with your data set. Settle in, this will be the longest part of the counting process.

### 3.1 Find candidate spots for each cell

In this step, Aro will find local brightness maxima for each image and make some (for now) black box calculations to describe each local maximum.

This will allow Aro to make guesses at which parts of the images could be spots to be counted. These guesses will become the candidate spots. This process is designed to include all possible spots regardless of whether or not each candidate is a real spot or not.

This process will create files of the form `dye_xxx_wormGaussianFit.mat` for each position and each dye. These files contain the stats for the candidate spots. These files will be saved in the `WormGaussianFit` folder.

In order to find candidate spots for all embryos, run the command `doEvalFISHStacksForALL` from your working directory.

### 3.2 Use Aro to classify the spot candidates

Aro uses the Random Forest algorithm to calculate the number of spots in each embryo. In order to do this, it has to first make a training set in order

to learn what a spot looks like in each dye (dye is also referred to as color channel). Then, using what it has learned from the training set, Aro classifies each of the candidate spots in the images as either a spot or not a spot in order to output a count of the number of spots in each embryo in each image (position).

### 3.2.1 Create a training set

The Random Forest algorithm needs user input to make the training set, i.e. the user has to give Aro examples of which spot candidates are actually spots. Some important guidelines to follow:

- Images between different data sets and even different dyes within the same data set can look very different. Make a new training set for each data set and each dye.
- In fact, Aro can't use training sets from one data set for another data set without producing errors later on, so don't reuse training sets.
- Good training sets will have both positive and negative examples, i.e. good examples of spot candidates that are spots and spot candidates that are not spots. Try to include as many positive examples as negative examples and also throw in a few ambiguous (in-between) examples that are hard even for a human to classify.
- Don't start right off with a huge set of training data. It's better to train with a few spots, observe the performance, and then make corrections and add those to the training set. Doing three or four rounds of this should be enough, and a training set that starts with around 300-400 spots for the first round of training is recommended.

1. From the working directory, run the MATLAB command

   `createSpotTrainingSet('dye_Posx','Probe_name')`.

2. In the `identifySpots` GUI window that pops up, either accept or reject spots (see Figure 5).
   - Keep the guidelines above in mind.
   - Especially remember to train spots across slices.
   - Do *not* touch the `Spot Rank` slider.
3. When finished, hit the `Done` button to close the `identifySpots`.

4. When `Finished` is pressed, a dialogue box will pop up asking if you want to continue training. Select `No`.

5. There should now be a `trainingSet_dye_Probe_name.mat` file in the `TrainingSets` folder.

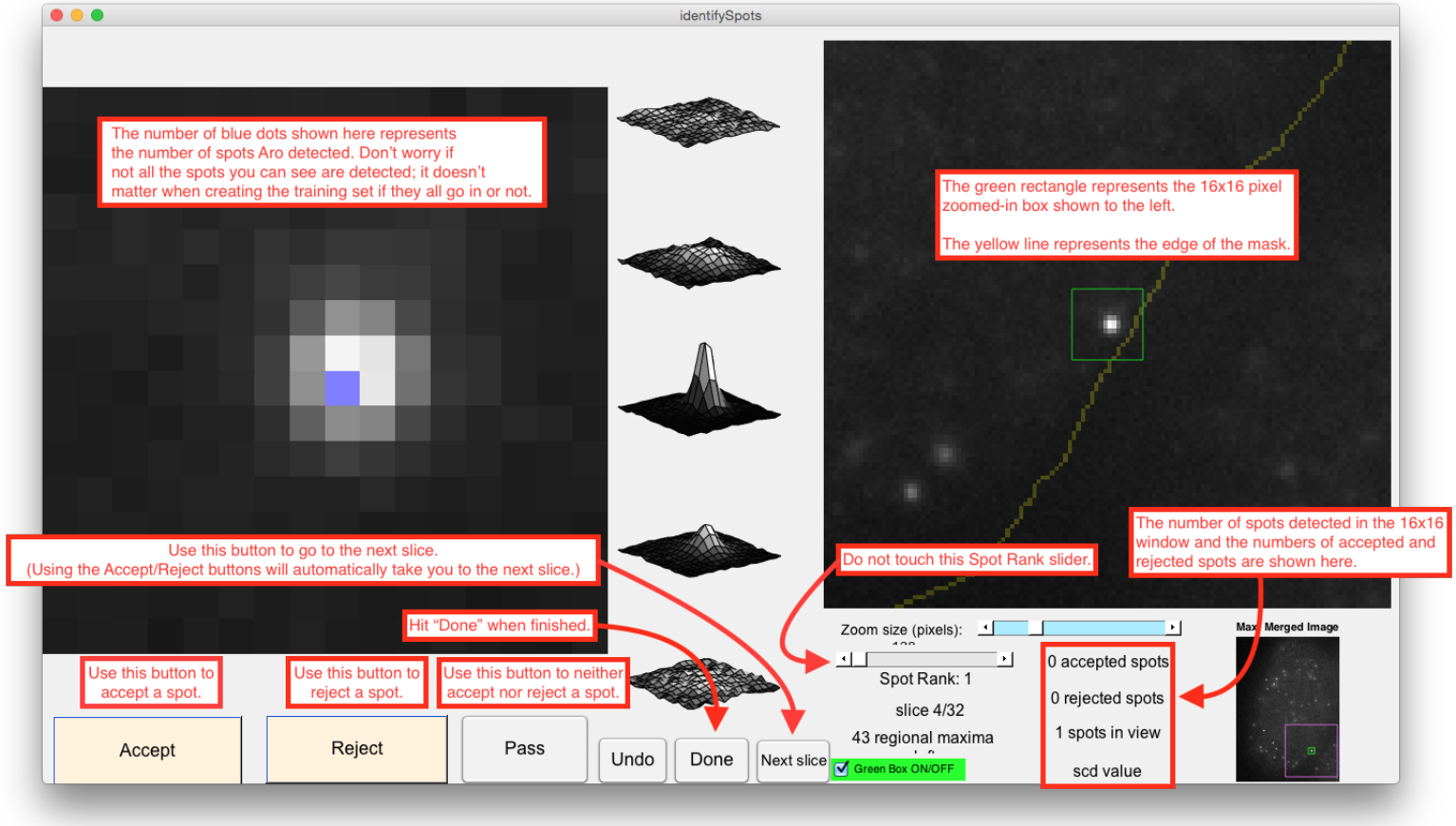


Figure 5: A labelled screenshot of the identifySpots GUI.

### 3.2.2 Use the training set to train a classifier

Now that a training set has been created, it can be used to train a classifier. The classifier is the part of Aro that, given a training set and a set of data about worms (remember the `wormGaussianFit.mat` file), will classify candidate spots as spots or not spots.

1. From the working directory, run the MATLAB command

   `load trainingSet_dye_Probe_name.mat` where dye is the name of a dye and Probe_name is the name of a probe.

2. From the working directory, run the MATLAB command

   `trainingSet=trainRFClassifier(trainingSet)`.

   - Don't be alarmed if this step takes a while. If you had 1000 training spots (you won't the first time), for example, this step could take 5 - 30 minutes depending on processing power.
   - At the end of this process, you should have an updated `trainingSet` file and a new `dye_Probe_name_RF.mat` file that contains the random forest data. The random forest data will also be in the updated `trainingSet` file.
   - Remember, these files will be in the `TrainingSets` folder (see Figure 1).

3. Repeat these steps until there is a training set for every dye.

### 3.2.3 Classify the spots using a training set

After a training set has been made, it can be used for an initial classification of the spots of a particular dye and probe. Don't expect this to be too accurate. Later, you will review this classification and make corrections to improve the training set and, by extension, the classifier.

1. From the working directory, run the MATLAB command

   `load trainingSet_dye_Probe_name.mat`. This naming scheme should be obvious by now.

2. In your MATLAB Workspace, you should see the variable `worms` and `trainingSet`.

3. From the working directory, run the MATLAB command `classifySpotsOnDirectory`.

- You could also choose to classify only a specific dye and specific position by using the command `classifySpots(worms, trainingSet)`. In order to run this command you would have to first run the commands `load trainingSet_dye_Probe_name.mat` and `load dye_Posx_wormGaussianFit.mat`.
- Another option is to still use `classifySpotsOnDirectory` but specify a training set and a dye. In order to do this, first load in the appropriate training set using the command given above. Then, from the working directory run the MATLAB command `classifySpotsOnDirectory(toOverWrite, trainingSet, dye)`. The parameter `toOverWrite` is a numeric boolean (1 or 0) that specifies whether the files should be overwritten or not and should of course be set to 1 to have the command do anything useful.
- It is recommended that instead of using specifics you just run the generic `classifySpotsOnDirectory` command. This will automatically classify all the spots for all the appropriate dyes and training sets for all the positions for you, which is what you want when you are reviewing the classification later.

#### 3.2.4 Review the classified spots and retrain

Use the `reviewFISHClassification.mat` GUI to review the spots classified by Aro. This step corrects classification errors made by Aro and gives you the option to add the corrections back to the training set. Don't try to correct all the errors at once — do three or four rounds of corrections.

1. From the working directory, run the MATLAB command `reviewFISHClassification(dye_Posx)`.
2. In the GUI window that opens, make corrections to the classifications (see Figure 6). If this is the first review, don't expect the classification to be very accurate.
   - Make sure that you are adding your corrections to the training set.
   - Make as many corrections as possible, but don't try to fix everything.
   - Remember to keep the guidelines for picking good training spots in mind.

3. When finished, press the `All Done` button. This will pop up a dialogue box asking "Retrain the classifier now?" to which you should respond by pressing the `Yes` button.

    - This process can take a while (remember how long training took the first time — there are even more spots now). *Do not close if it looks like it's not doing anything! Check the MATLAB Command Window — it is probably still running the training step.*
    - You can also retrain without pressing `All Done` as shown in Figure 6, but this is not recommended because it will probably just slow you down unnecessarily (you should be redoing and rechecking the classification after each retraining anyway).

4. Repeat this process until the classifier does not appear to show improvement (approximately 3 - 4 times).

5. Repeat the retraining procedure for all the classifiers (i.e. each dye/probe). Try to use multiple positions when retraining.

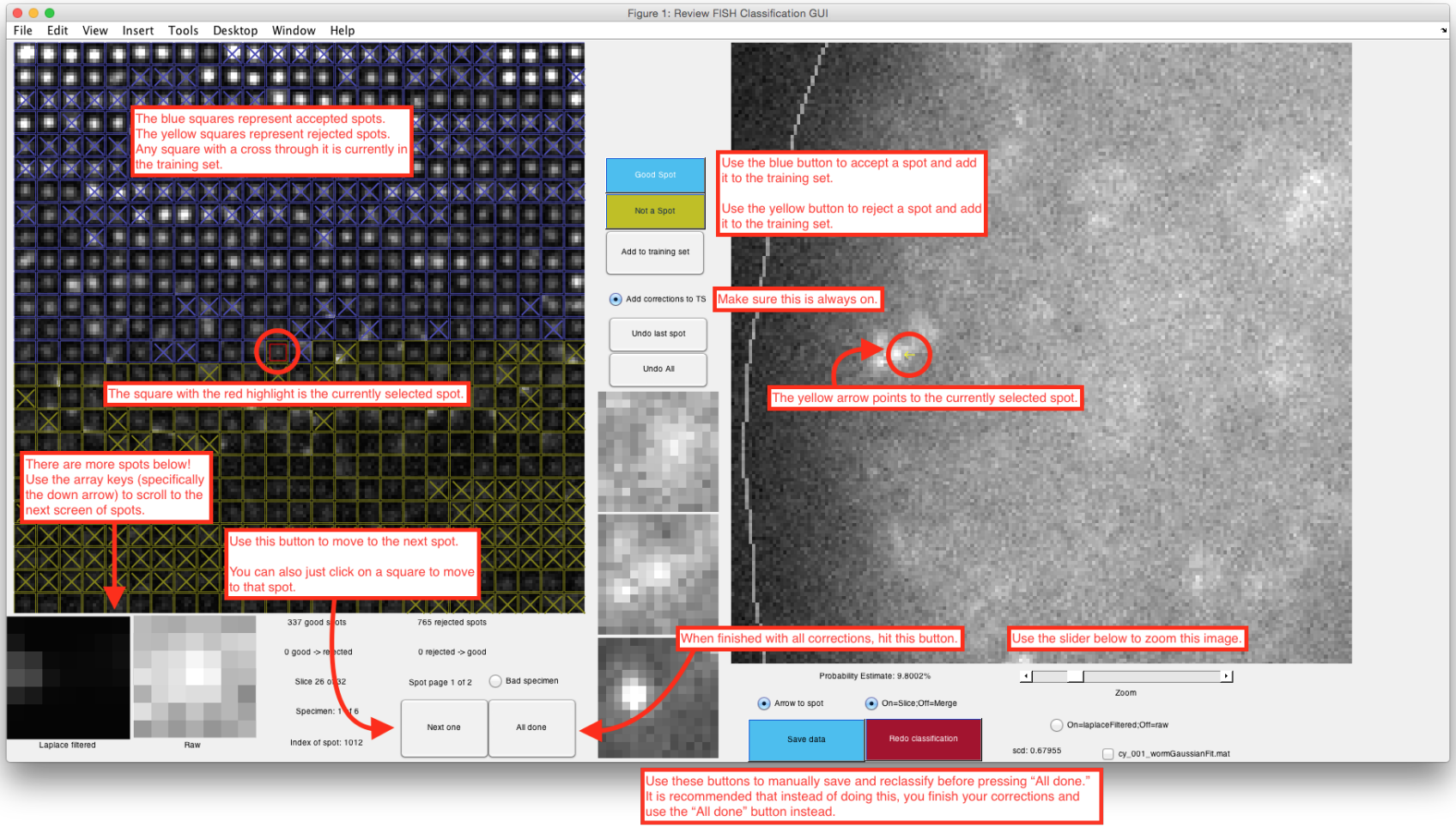


Figure 6: A labelled screenshot of the review GUI.

### 3.2.5 Classify all spots automatically

Once the classifiers have been sufficiently trained and retrained, it is time to let Aro count all the spots. Remember, each classifier is for one type of dye in the same data set, simply different positions.

1. From the working directory, run the MATLAB command `classifySpotsOnDirectory`.
2. All the candidate spots for all the embryos should now be classified.
3. Check to make sure there are newly updated `spotStats.mat` files for all the possible positions for all the dyes in the `SpotStats` folder.

### 3.2.6 Sum all the spot counts

Aro now has the spot counts for each embryo at each position for all the dyes used. This step will give a summary of all the spot count data.

1. From the working directory, run the MATLAB command

   `spotStatsDataAligning2(suffix)`.

   - The parameter `suffix` is an arbitrary file suffix that will be used to name the resulting file of this command.
   - It is recommended that the suffix be a date in the form of a string of integers, i.e. `yyyymmdd` and e.g. `20151031`.
2. This command should produce two files: `wormData_suffix.mat` located in the `AnalysisDir` folder and `ErrorPercentagePlot_suffix.fig` located in the `Plots` folder. For a reminder about the directory structure, see Figure 1.

# 4 Reading Results From Aro

By now, the counting should be completed. You can now run some commands to access the completed counts data.

## 4.1 Statistics for each position

For any position, you can manually open up the `spotStats` file by using the MATLAB command `load dye_Posx_spotStats`. This will allow you to run the command `spotStats{n}` which will display output similar to Figure 7.

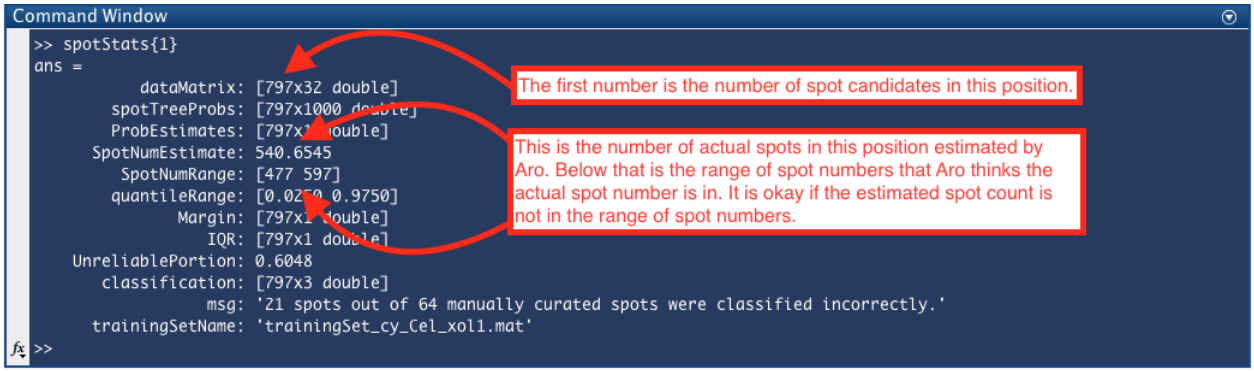


Figure 7: A labelled screenshot of the output from a `spotStats` file for a specific dye and position.

## 4.2 Using the summarized data file

However, you should have already run the command `spotStatsDataAligning2`. This means there is a `wormData` file that contains summaries of relevant data for each worm and each object at each position (which should mean each embryo at each position for all the dyes used). In order to access this file, from the working directory run the MATLAB command `load wormData_suffix`. Now, there is a MATLAB struct array with fields `spotInfo`, `dye`, `dye1`, ..., `dyen`, `errorPercentage`, and `meanRange` in the Workspace.

The field `spotInfo` contains information about each object (which should be each embryo — regardless of the dyes used, since each dye should have each object for all positions). For each object, `spotInfo` contains three numbers that represent, in order: the overall number (index) of the object, the position that the object is located in, and the number (index) of the spot *in that position*. Running the MATLAB command `wormData(1).spotInfo` will return a matrix containing these three values for all the objects, but this can be overwhelming when there are hundreds of embryos. Running `wormData(1).spotInfo(n,:)` will show the spot information for the $n^{\text{th}}$ object.

The field `dye` contains a list of the dyes used as defined in `Aro_parameters.m`. The other fields that are named by each specific dye contain information about the spot counts for each of those dyes. In one of those fields, the following numbers can be found for each object (which should be each embryo): the number of the position in which the object is located, the es-

timated spot count for that object, the upper error, and the lower error. Running the MATLAB command `wormData(1).dye`$n$ will return a matrix containing these four values for all the objects. This can be overwhelming if there are a large number of embryos (just like the `spotInfo` field but there is even more data since each dye has its own field). Running `wormData(1).spotInfo(`$n$`,:)` will show the spot count estimate and other data for the $n^{\text{th}}$ object.

The field `errorPercentage` contains the error percentage of the spot estimate for each object and each dye. This value is calculated as

$$\frac{\frac{U+L}{2}}{\text{total spot number}} \cdot 100\%. \tag{1}$$

$U$ represents the upper error and $L$ represents the lower error. In order to get the error percentage for the $n^{\text{th}}$ dye and $k^{\text{th}}$ object, run the MATLAB command `wormData(`$n$`).errorPercentage(`$k$`)`.

The field `meanRange` contains the average of the difference between the upper and lower bound spot estimates (the average range) from all the objects for each dye. In order to get the mean range for the $n^{\text{th}}$ dye, run the MATLAB command `wormData(`$n$`).meanRange`.

With each of these fields and their corresponding commands, you can access the spot count data for each embryo. It is also possible to load and view this data using the MATLAB GUI by just loading the `.mat` file in MATLAB through its file browser and viewing the variable from the MATLAB Workspace. However, once you get used to it, it will be faster to just run the commands. See Figure 8 for examples of the outputs from each of these fields.


Figure 8: A screenshot showing example output from the various fields in `wormData` using the fourth object and the second dye.

### 4.3 Accessing and interpreting error plots

When the `spotStatsDataAligning2` command was run, it produced a `.fig` file. This file is a plot of the error percentages that can be found in the `wormData` file (see Section 4.2). The plot shows error percentage versus number of spots for each object in each dye, color coded by dye. In order to view the figure, from the working directory in MATLAB, run the command `open AnalysisDir/Plots/ErrorPercentagePlot_suffix.fig`. This will open up the plot in a MATLAB figure window. See Figure 9 for an example of what to expect from this plot. When the classifiers are well trained, you should see that the error percentage plot changes from a higher average error percentage to a lower error percentage for any given object. This means a poorly trained classifier will have a more globular, scattered error percentage plot while a well trained classifier will have a more flat, clustered error percentage plot. A well trained classifier also has lower mean ranges (found in `wormData(`*n*`).meanRange`). These plots are of course not directly showing you that accuracy is improving since Aro has no idea what the proper counts actually are. However, both the error percentage plot and the mean range values should give you an idea of whether the classifier is becoming stronger.

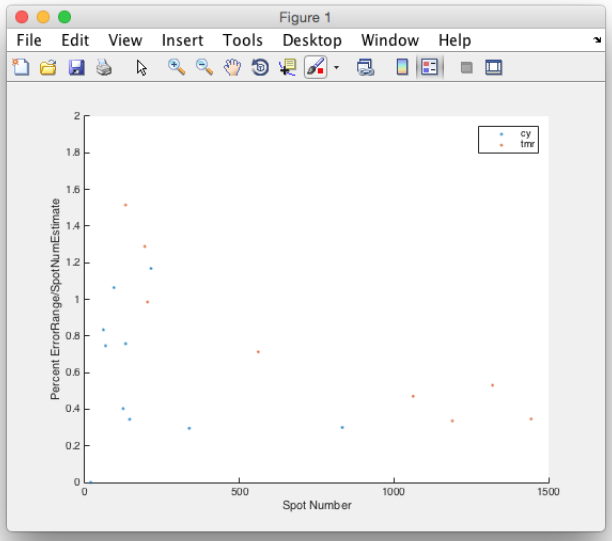


Figure 9: A screenshot of an error plot produced with example data.

# 5 Automatically running Aro on PACE

In order to use Aro on PACE, follow the steps below. This assumes that a training set has already been produced according to the instructions in Section 3.2.1. We also assume basic knowledge of `bash` commands.

1. First, `SSH` into PACE using `user@biocluster-6.pace.gatech.edu`.

2. In your `data` folder, place a copy of Aro using `scp -r AroSpotFindingSuite user@biocluster-6.pace.gatech.edu: /data/` from your local machine (this assumes you have already obtained a copy of Aro from <https://github.com/diptodip/AroSpotFindingSuite/archive/master.zip> and that you are currently in the folder containing the Aro folder).

3. In your `data` folder, construct a new Aro project folder as detailed in Figure 1. To do this, use the `mkdir` command. Place all of the `*.pbs` script files in the Aro folder in this project folder using `cp`. Also place the training set files you have produced in the training set folder.

4. Copy over the `Aro_parameters.m` example file from the Aro folder (and make sure that it is filled out properly for your analysis).

5. Submit the Aro job to the cluster queue by using `qsub full_run_aro.pbs`.

   Alternatively, run the steps only up to masking and producing thumbnails using `qsub automask_pace.pbs`.

   Then, copy over the `SegmentationMasks` folder (as well as `Aro_parameters.m`) to your local machine using `scp -r user@biocluster-6.pace.gatech.edu: /data/<project_fo ./` on your local machine (this will place the folder in your current folder on your local machine).

   Open MATLAB to the directory containing the `SegmentationMasks` folder and run the command `show_mask_preview` to preview the masks as shown in Figure 3.

   Back on PACE, finish the remainder of the analysis using `qsub analysis_after_masks.pbs`.

6. The output of the analysis will show up as `output.csv` and the Aro log/console output will be recorded in the `*.output` file.

**This concludes the Aro guide.**