

Particle Separation Using PS-POLY in Igor Pro

E.A. Conley^{1,2}, G. M. King Laboratory¹

1] University of Missouri, Department of Physics and Astronomy

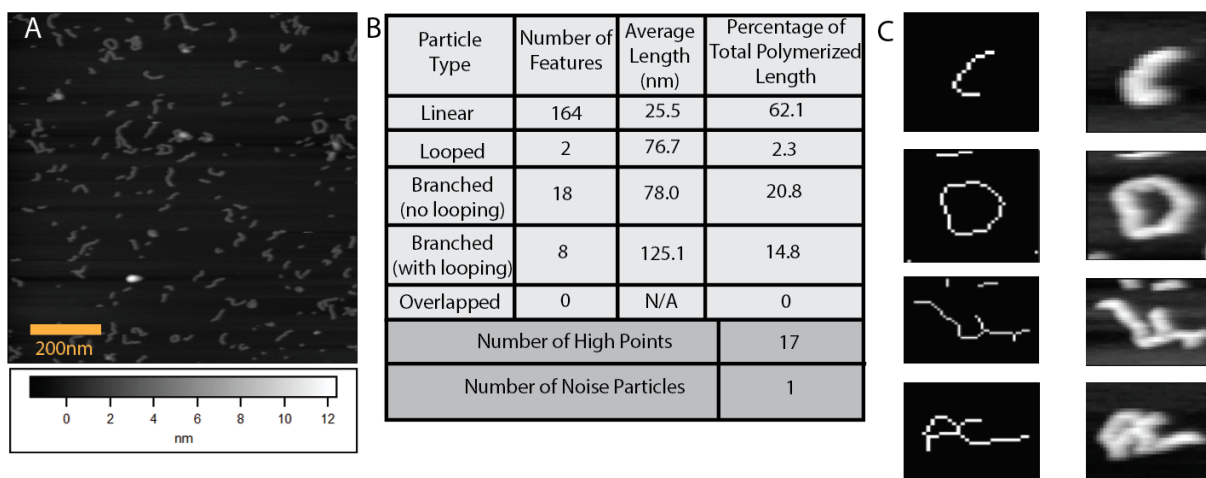
Email: kinggm@missouri.edu

2] Washington University in Saint Louis, Department of Neuroscience

Email: elizabethconley@wustl.edu

Introduction

The PS-Poly algorithm is a particle detection program designed to separate individual features based on shape using images taken using Atomic Force Microscopy. The program sorts particles into the following groups: linear, looped, branched without looping, branched with looping, overlaps, and noise. For linear molecules, persistence length is calculated using the Worm-Like Chain model. Due to an interpolation that is performed to increase the pixel density of the image, the persistence length result is achieved with subpixel precision.



A The original image of candidalysin. **B** The distribution of shapes that image A was sorted into. **C** Examples of the different shape types. Showing linear, looped, branched without looping, and branched with looping from top to bottom respectively.

The output may vary slightly depending on the scaling factor you select for the interpolation that increases the pixel density, and the value of the height threshold you select when prompted by the program. For consistent results, the threshold can be applied to the images manually.

In this tutorial, we walk through the use of the PS-Poly algorithm with our provided software, written in the scientific analysis software Igor Pro (<https://www.wavemetrics.com/>). No prior experience with Igor Pro or coding is required, we will proceed in baby steps.

We hope you find success using the Hessian blob algorithm and our provided software! Please do not hesitate to contact us with questions, suggestions, or bugs.

Contents

I.	Preliminaries	3
II.	PS-Poly shape categorization	9

I. Preliminaries

In this tutorial, we go through the basics of loading images into Igor Pro 7 for analysis and creating a data folder for the loaded images. We will also review some of the basic concepts and language used in the Igor Pro environment. Launching Igor Pro will begin what is called a new Igor “experiment”. This is a moniker for a new file, which will hold all of our data and the results of our analysis. There are three main types of data in Igor, these are variables, strings, and waves. Variables are numbers that we store and name on the computer, such as the variable $a = 3$. Strings are similar, but instead of numbers they are just sets of characters, such as `str = "Hello"`. We may use these variables to store information, perform calculations with them, or change them as we desire. Finally, waves are collections of variables or strings. Most often, we deal with numeric waves, which hold sets of numbers in a simple array. These waves might hold one-dimensional data arrays such as time-series data, two-dimensional data such as images, or even three and four-dimensional volumetric data.

The two main components of Igor we'll need for the tutorial are the data browser and the command line. In an Igor experiment, all data (including waves, variables, and strings) is organized in the data browser folder system. In the data browser, we are able to view our data and add or delete data folders to keep our analysis organized. The command line, similar to any other programming environment, is our interface with Igor where we execute commands to run our analysis, edit waves and variables, or execute any other Igor command.

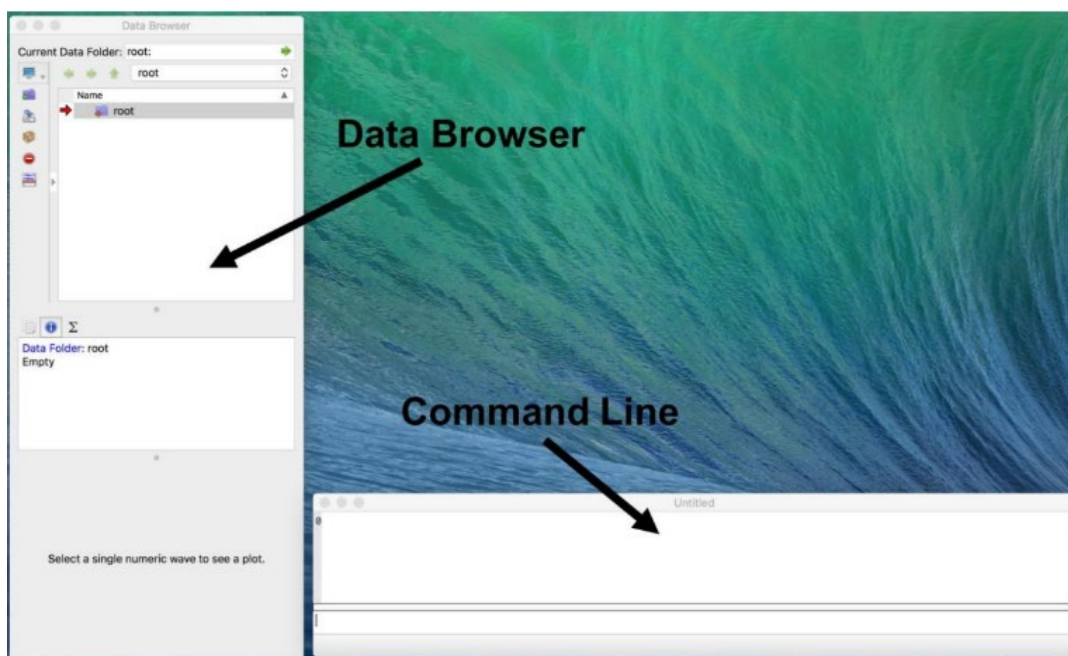


Figure 1: The Igor interface, including the data browser and command line. Note that if you are running Igor on a Windows machine, the entire experiment will appear in its own window. On a Mac (pictured here) the various Igor windows will all appear floating on your desktop.

Before we load any images into our experiment, let's create a new data folder to store them in. To create a new data folder, simply hit the new data folder button in the data browser and name your folder appropriately.

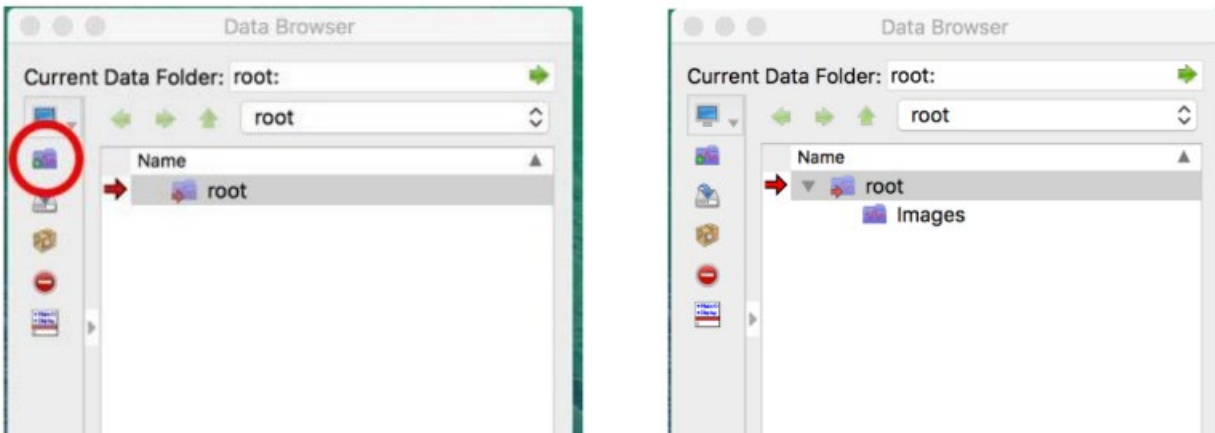


Figure 2: Creating a new data folder. In the data browser, click the new data folder button shown circled in red and name you data folder, here named “Images”.

Your new data folder will appear beneath the root data folder, indicating that is inside the root data folder as everything will be. You'll notice that a red arrow points to the root data folder, this is an important point which indicates that the root data folder is the “current” data folder. It is important to keep track of which data folder is the current data folder, as any new or loaded data will automatically be put into this data folder. You can change the current data folder by right-clicking on any data folder and selecting “Set Current Data Folder”.

Let's now load our images. Under the data tab at the top of the screen, we will need the “Load Waves” section. The option we choose in the “Load Waves” section will depend on how your images are stored. In order to perform our analysis, we will need to have the images in an Igor friendly format, AKA waves. If your data is already saved in native Igor format as ibw files, choose the Load Waves → Load Igor Binary option. If you images are stored in a standard image format such as tiff, jpeg, or png, choose the Load Waves → Load Image option. If your data is stored in a simple spreadsheet style such as Excel format or csv, you'll need the Load Waves → Load General Text or Load Delimited Text options.

After importing your images into Igor, they will appear in the current data folder. Let's move them into our Images data folder by simply dragging and dropping.

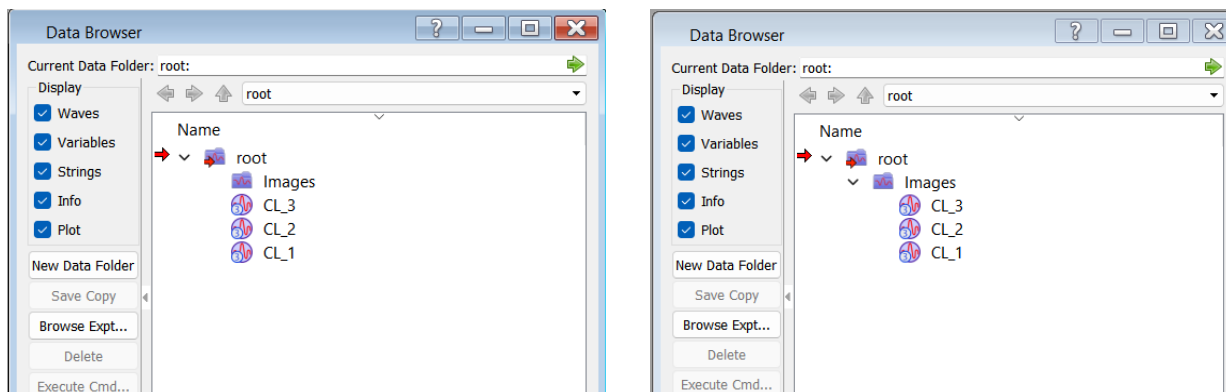


Figure 3: Loading images and moving them into a data folder.

To view our images, we may either right click on them and select “New Image” or use a command line function. To view an image using the command line, we need to introduce the concept of the wave path.

Every wave, such as our images, has both a name and a path. The name simply identifies the wave, whereas the path specifies the full location of the wave in the data browser. For example, here we loaded three waves, one of which is called “CL_1”. The full path of this wave “root:Images:CL_1”.

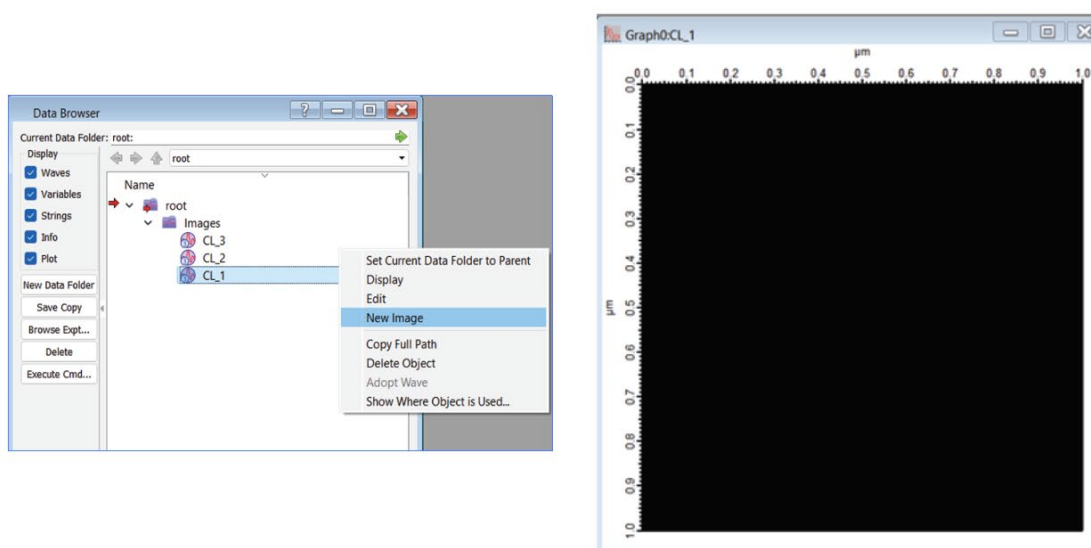


Figure 4: Viewing an image by right clicking and selecting “New Image”.

Next we want to scale the image so we can view the particles in it. To do this, right click on the image, and press “Modify Image.” Then we will check the boxes marked “Autoscale only visible layer” and “Autoscale only visible XY.” After that, we press “Do it,” and we will be able to see the image of Candidalysin.

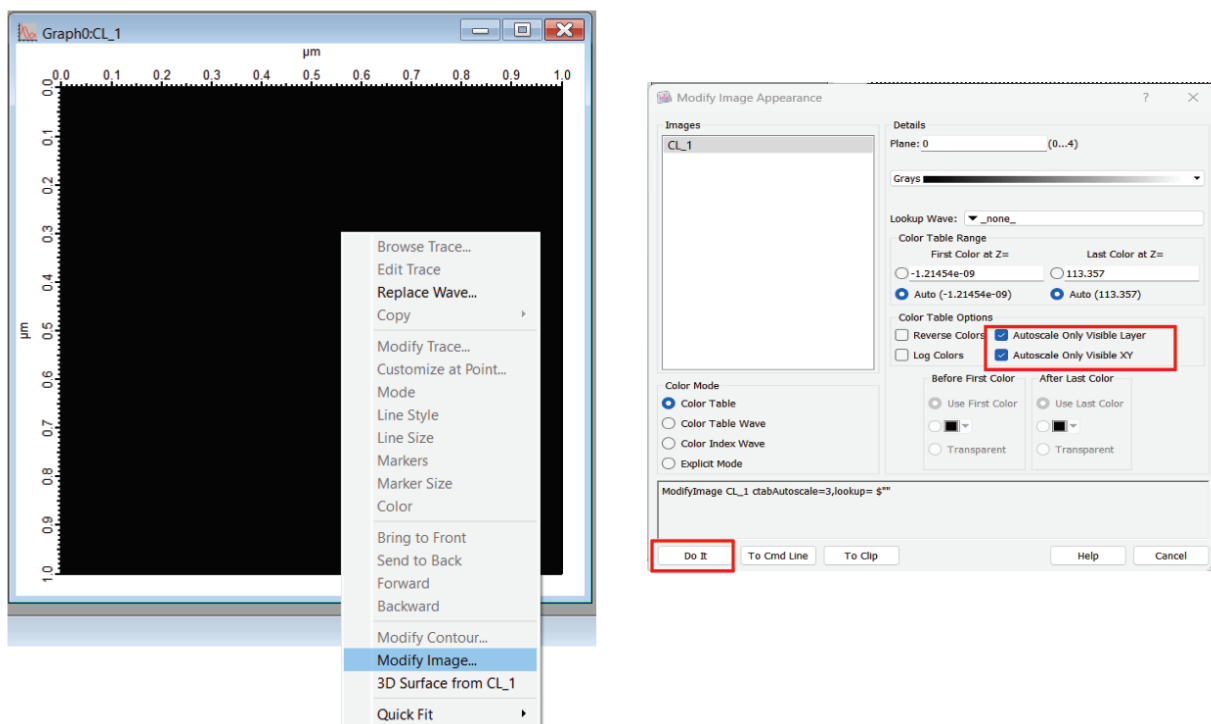


Figure 5: Scaling the image by using the Modify panel

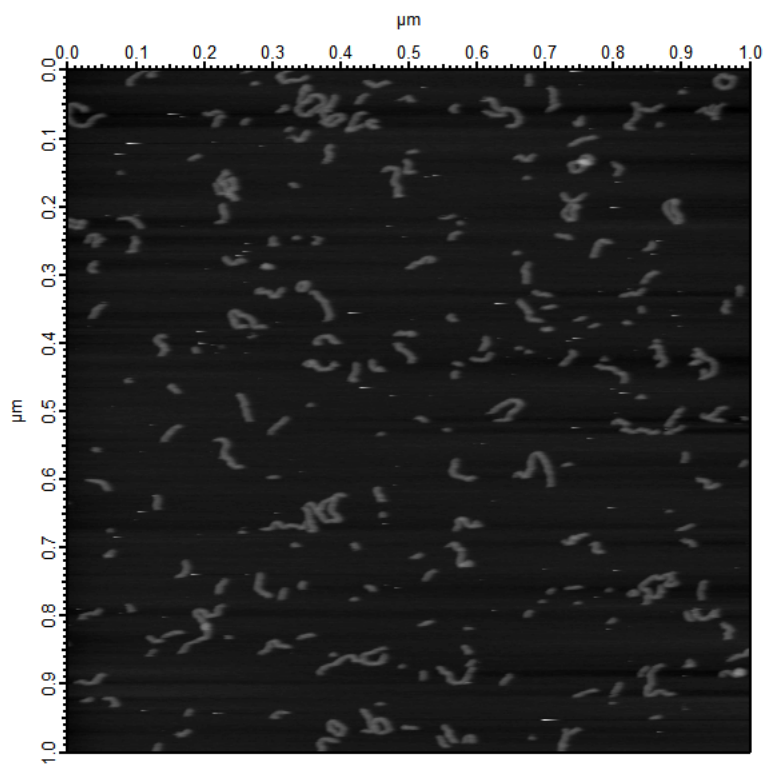


Figure 6: The image CL_1 after scaling

To view an image using the command line, we use the Igor function NewImage. To execute the command, we type it into the command line along with the full path of the image we want to display, so Igor knows exactly which image you are talking about. For example, to view the CL_1 image we type the following into the command line.

```
NewImage root:Images:CL_1
```

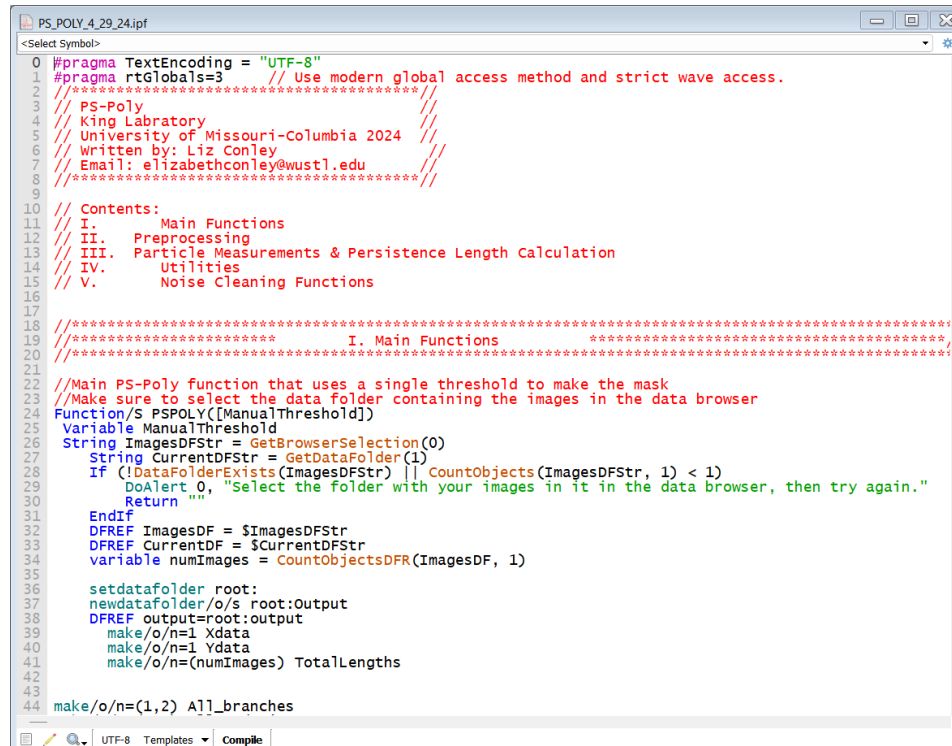
Note that Igor automatically inverts the y-axis of the image. To view the image with an upward oriented y-axis, use the /F flag in the NewImage command:

```
NewImage/F root:Images:CL_1
```

Also note that the NewImage function only works for two and three dimensional data, not one dimensional data. To view one dimensional data, we right click and choose the “Display” option or use the “Display” command analogously on the command line.

Next, we need to load what is called a procedure file, these are the codes we write to extend Igor’s capabilities and design our own analysis procedures. The PS-Poly algorithm is provided in a single procedure file which should be downloaded from our website, which should be named PS_Poly_Vx.ipf. In Igor, go to

File → Open File → Procedure to load the procedure file. Once loaded, it should look something like this.



```
0 #pragma TextEncoding = "UTF-8"
1 #pragma rtGlobals=3 // Use modern global access method and strict wave access.
2
3 PS-Poly
4 King Laboratory
5 University of Missouri-Columbia 2024
6 Written by: Liz Conley
7 Email: elizabethconley@wustl.edu
8
9
10 // Contents:
11 // I. Main Functions
12 // II. Preprocessing
13 // III. Particle Measurements & Persistence Length Calculation
14 // IV. Utilities
15 // V. Noise Cleaning Functions
16
17
18 //***** I. Main Functions *****
19 //*****
20 //*****
21
22 //Main PS-Poly function that uses a single threshold to make the mask
23 //Make sure to select the data folder containing the images in the data browser
24 Function/S PSPOLY([ManualThreshold])
25 Variable ManualThreshold
26 String ImagesDFStr = GetBrowserSelection(0)
27 String CurrentDFStr = GetDataFolder(1)
28 IF (!DataFolderExists(ImagesDFStr) || CountObjects(ImagesDFStr, 1) < 1)
29 DoAlert 0, "Select the folder with your images in it in the data browser, then try again."
30 Return ""
31 EndIf
32 DFREF ImagesDF = $ImagesDFStr
33 DFREF CurrentDF = $CurrentDFStr
34 variable numImages = CountObjectsDFR(ImagesDF, 1)
35
36 setdatafolder root:
37 newdatafolder/o/s root:Output
38 DFREF output=root:output
39 make/o/n=1 Xdata
40 make/o/n=1 Ydata
41 make/o/n=(numImages) TotalLengths
42
43
44 make/o/n=(1,2) A[]_branches
```

Figure 7: The loaded PS-Poly procedure file

All of the functions needed for PS-Poly analysis may be found here. To load all of the functions into Igor, you may need to hit the “Compile” button which will be present in the bottom of the procedure file. If it is not there, don’t worry it has already been compiled. After compiling the code, all of the user-defined functions we provide have been loaded and you may minimize the window if you like.

II. PS-Poly Shape Categorization

In this tutorial, we apply the PS-Poly particle sorting algorithm to a set of loaded images. To begin, load a set of images and place them in a data folder as covered in tutorial I.

We now run the PS-Poly algorithm on the images, but we must first identify for Igor the data folder containing our images. Highlight the data folder containing the images by single clicking on the data folder, as shown below.

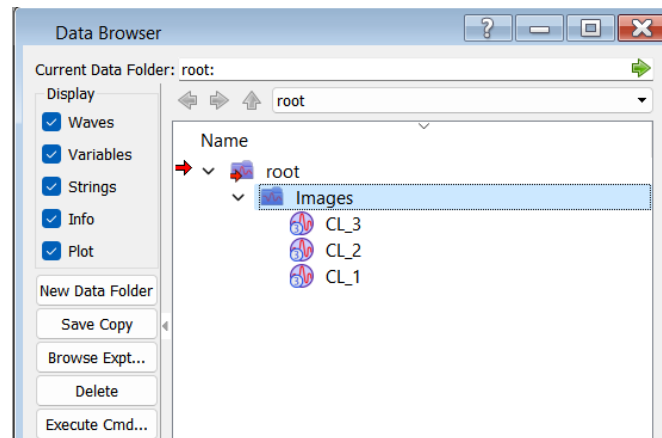


Figure 8: The highlighted data folder containing the images to be analyzed.

With the correct data folder highlighted, we now execute the PSPOLY command to run the algorithm on all of our images. To get the same results, we will select our threshold manually at 1.5 nm. To do this, type the following into the command line.

```
PSPOLY(ManualThreshold=1.5e-9)
```

Using this command on this set of images should print a persistence length result of 10.0 +/- .4 nm. The output graph showing the persistence length fit to the worm like chain model equation using the root-mean-square of the end-to-end distances on linear particles and their contour lengths is shown in figure 9.

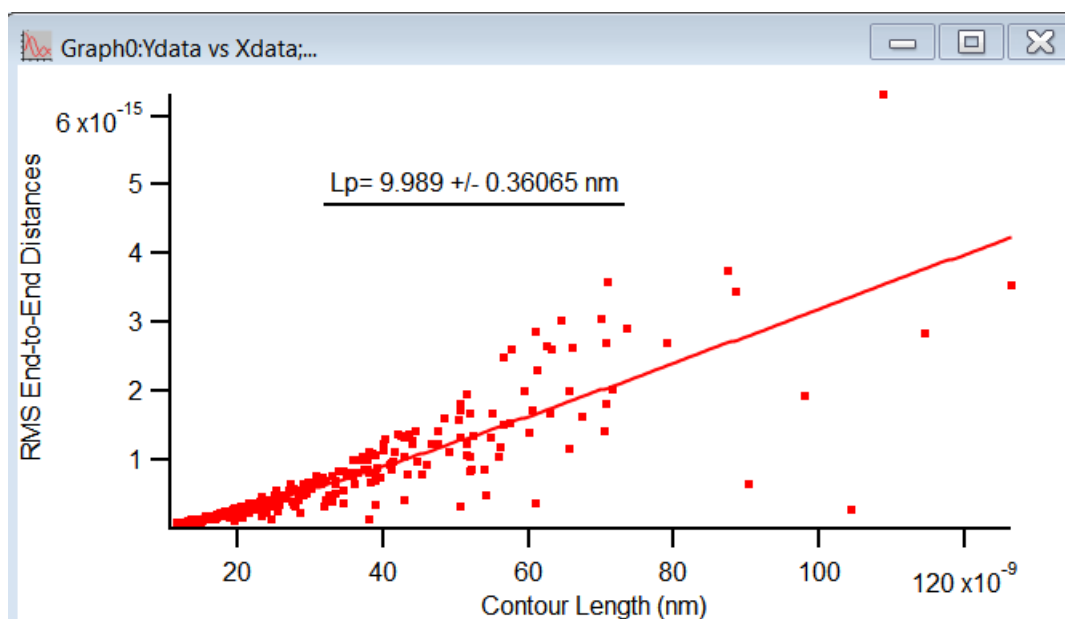


Figure 9: Output graph showing the data fit to a persistence length of $10.0 \pm 0.4 \text{ nm}$

To use the interactive threshold option, simply type the following into the command line, with the images data folder still selected:

PSPOLY()

Using this option, you will need to select a threshold for each image in the folder.

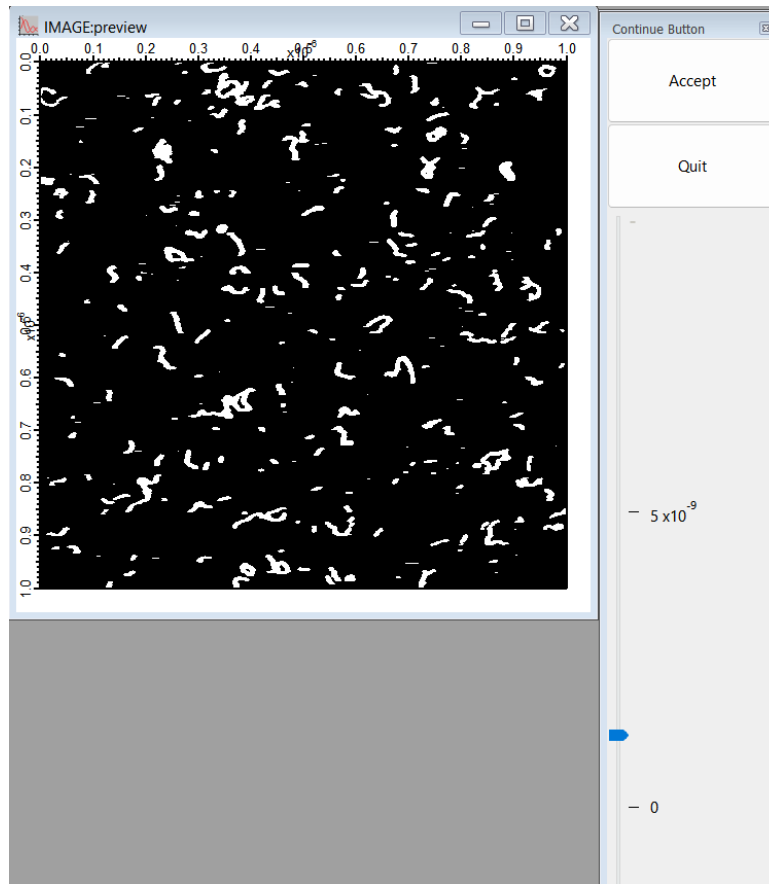


Figure 10: Interactive slider to select threshold value.

Output

In addition to the persistence length output graph, there will be a new folder in your data browser titled “Output.” This is where all the information about the particles is stored. When you open the folder, it should look something like figure 11.

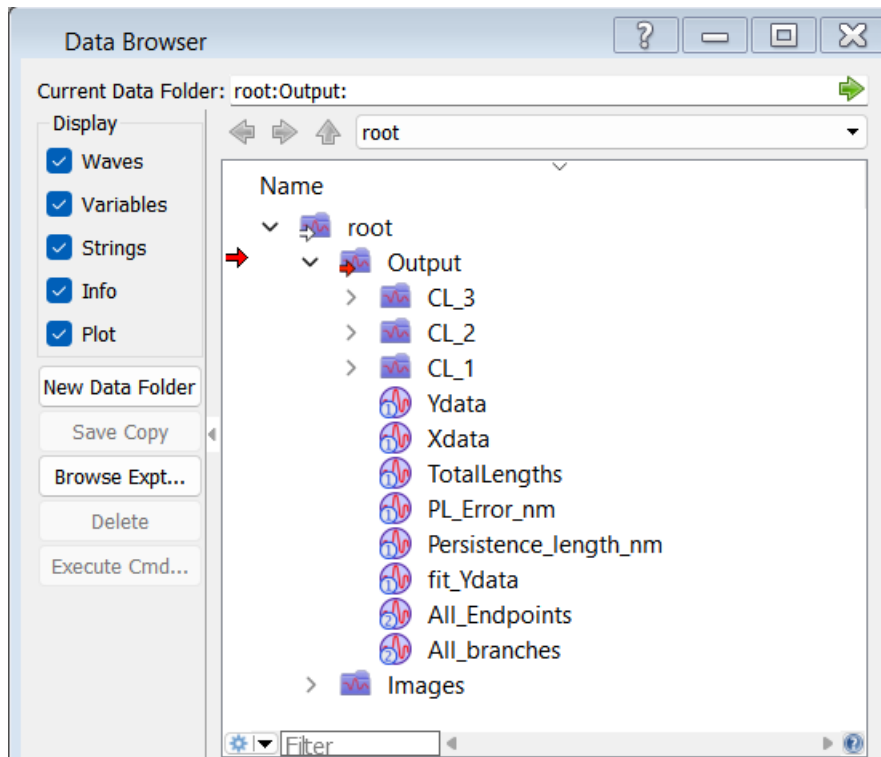


Figure 10: Output folder produced by PS-Poly

The output folder contains a subfolder for each image, as well as waves titled “Ydata”, “Xdata”, “TotalLengths”, “PL_error_nm”, “Persistence_length_nm”, “fit_Ydata”, “All_Endpoints”, and “All_branches.” The contents of each wave are the following:

Ydata: The root-mean-square of the end-to-end distances on the particles sorted as linear. This is used for the persistence length estimate.

Xdata: The contour lengths of linear particles whose index value is the same as it’s corresponding point in YData. This is also used to get persistence length.

TotalLengths: A list containing the combined lengths of every particle in each image in the data folder used on the program.

Persistence_length_nm: The persistence length value for all of the images

PL_error_nm: The error in the persistence length value for all images

fit_Ydata: This wave corresponds to the output graph shown in figure 9 which is the persistence length fit when plotting Ydata vs. Xdata

All_Endpoints: A list containing the endpoint coordinates for all the images.

All_Branches: A list containing the branch point coordinates for all the images.

To view the values for each of these waves, double click on them. A matrix should appear that looks like the one below.

[illegible]

Figure 11: The total polymerized length for each image, stored in the TotalLengths wave

Now lets see what's in the folders for each of the images. Click the arrow next to the "CL_1" folder, and you should be able to see all of its contents.

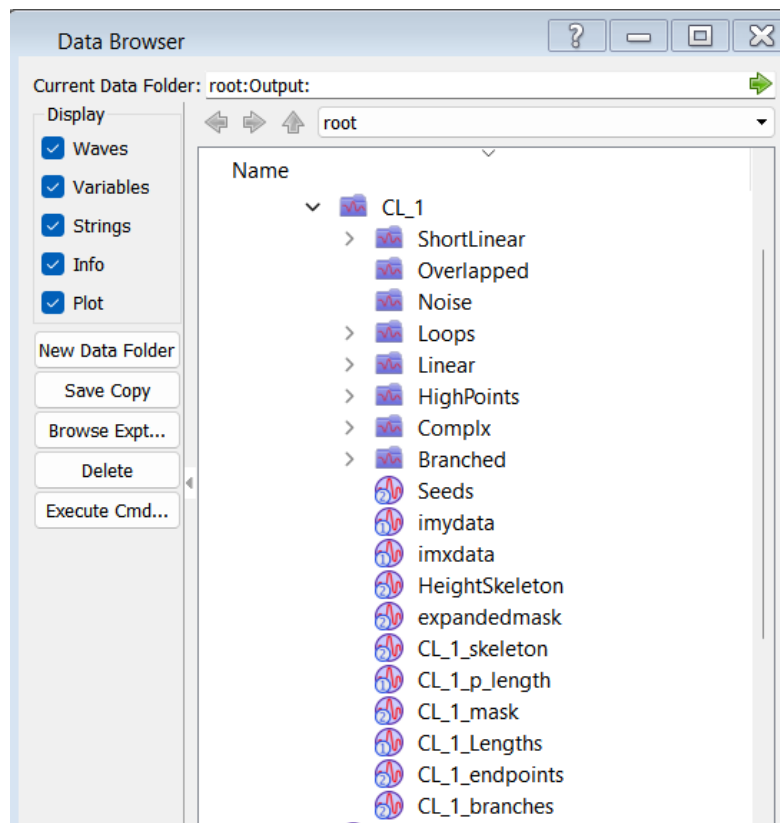


Figure 12: The CL_1 subfolder within the main Output folder

There are 8 subfolders and 11 waves in the image data folder. Here's a key to what each of them are:

imydata: The root-mean-square of the end-to-end distances of the linear particles in the image CL_1. This can be used to find the persistence length of only the particles in the CL_1 image.

imxdata: The contour lengths corresponding to the end-to-end distances represented in imydata.

HeightSkeleton: This is the skeletonized image that retains the original height information from the CL_1 image.

ExpandedMask: This is the floodmask that is created with an increased pixel density

CL_1_skeleton: The skeletonized image of CL_1

CL_1_p_length: A wave which holds the persistence length estimate for the linear particles in the CL_1 image.

CL_1_mask: This is the floodmask that is created at the user specified threshold. Pixels in the image that have a value above the threshold become equal to 1, and values below the threshold become equal to zero.

CL_1_Lengths: A list of the contour lengths for every particle in the image

CL_1_endpoints: The coordinates of all of the endpoints located in the CL_1 image

CL_1_branches: The coordinates of all of the branch points located in the image.

The 8 subfolders correspond to sorting categories for the individual particle information. Particles are sorted by the following criteria.

Linear: Particles that have 2 endpoints and no branch points

ShortLinear: Linear particles which are less than 5 pixels in length

Loops: Particles that have no endpoints and no branch points

Branched: Particles that contain endpoints and branch points that do not loop

Complex: Particles that branch and also loop

Overlapped: Particles with a branch point that has a height which is over 1.5 times the average height of the rest of the polymers in the image

Noise: Particles in which over 80% of the pixels are over 1.5 times the average height of the rest of the polymers in the image

HighPoints: Particles that contain a pixel which is over 1.5 times the average height of the rest of the polymers, but does not meet criteria to be sorted as overlaps or noise. These particles are copied into the HighPoints folder, meaning that the particle information is still sorted into one of the other folders.

Lets open up one of these folders to view the individual particle information. Below shows the data browser after opening the “Loops” subfolder.

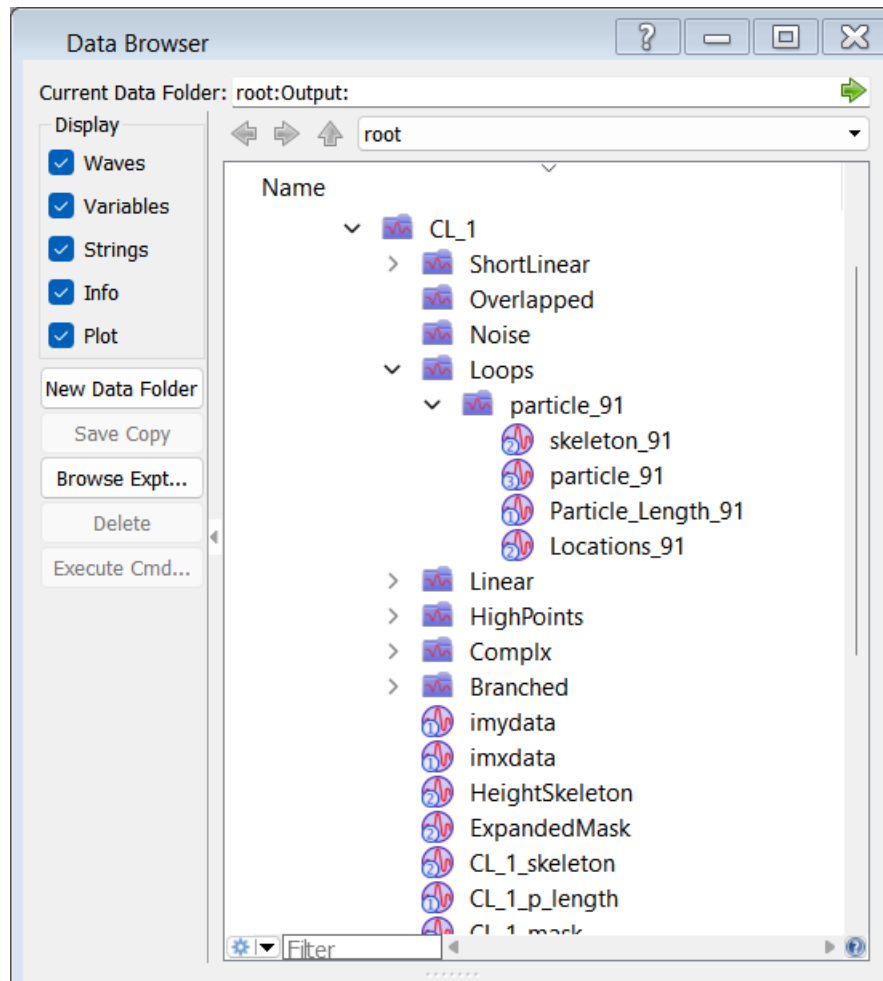


Figure 13: The Loops subfolder located within the CL_1 subfolder

Sorted particles each have their own data folders. For particle 91 shown in figure 13, there are 4 waves.

skeleton_91: refers to the skeletonized particle

particle_91: cropped particle 91 from the original image of CL_1

Particle_Length_91: holds a value corresponding to the contour length of that particle

Locations_91: contains a list of coordinates corresponding to particle 91 on the skeleton

To view the particle, right click on the “particle_91” wave and then press “New Image.” After using the modify panel to autoscale the image, it should look something like figure 14.

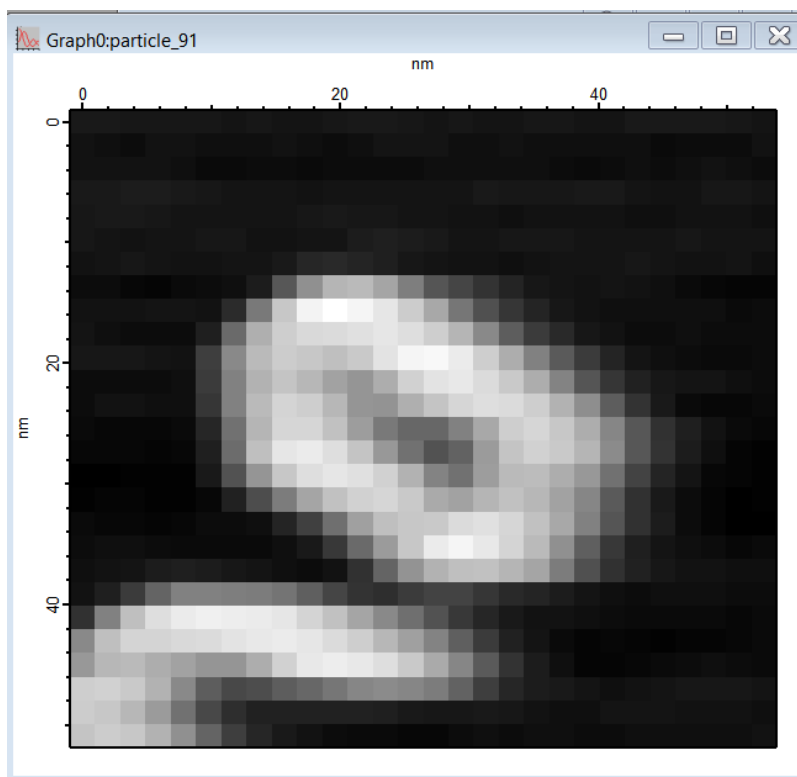


Figure 14: The sorted particle 91 image

To save multiple output folders from PS-Poly, rename the folder “Output” before running PS-Poly on a new folder. You can rename the folder by selecting the output folder in the data browser, then right clicking on it.