MDPI

Article

Developing Data Analysis Tools for Zebrafish Data in IMARIS

Catherine Kung[1,*], Summer Yu[1,*], Haley Schmidt[1,*], Carol Milton[1,*]

[1] Smith College - SDS department 1 Chapin Way, Northampton, MA, USA;
* Correspondence: ckung@smith.edu; Tel.: +1-413-801-1410. syu@smith.edu; Tel.: +1-413-404-7578. heschmidt@smith.edu; Tel.: +1-651-315-2760. ctmilton@smith.edu; Tel.: +1-508-930-5791.

Version December 15, 2021 submitted to Journal Not Specified

Simple Summary: Using IMARIS to visualize embryo development of zebrafish

Abstract: The Barresi Lab at Smith College recently acquired a light-sheet microscopy system through a grant from the Arnold and Mabel Beckman Foundation. The lab is interested in studying bio-electric patterns in embryonic development of zebrafish. To achieve this goal, we utilize the 3D images from this microscope in Z-stack format by using Imaris as a major analytical program. Although it is an excellent tool for 3D image analysis, the lab requires more tailored processes to make computations more efficient. To initiate the development of such computations, we create two python plugins for Imaris. One aims to transform intensity of data above a certain threshold into a new intensity level. Another is to count the number of surfaces and output it as a text file. These plugins ultimately streamline the process of analyzing incoming data from the light-sheet microscopy system and may facilitate future analytical processes for future projects.

Keywords: Imaris; Plugin; Automate

1. Introduction

The Barresi Lab at Smith College is a developmental biology lab led by Professor Michael Barresi. They use zebrafish as a model system to investigate the molecular and cellular mechanisms which control how the nervous system is built. The lab's research is divided into three projects to address how the neurons are created during embryonic development, how the nervous system gets wired, and what sorts of environmental factors impact neural development. Through a grant from the Beckman Center for Advanced Light Sheet Microscopy and Data Science, the lab acquired a new light-sheet microscope which sends 3D images of zebrafish embryos to be analyzed in Imaris, a microscopy imaging analysis software. Our group is working with the lab to achieve the following goals: 1) Use Imaris to develop data analysis tools which help the lab further their research. 2) Document protocols for the process of going from data aggregation to data utilization. 3) Create plugins for Imaris to automate data analyses on 3D and 4D images. As our analytical software for this project, Imaris is an Interactive Microscopy Image Analysis software offered by Bitplane that allows users to process and analyze fluorescent 3D and 4D microscopy images [1]. Although previously the lab had other software that allowed for 3D visualizations, it didn't have as high of a resolution as Imaris, therefore the lab has decided to switch software. However, they are not as familiar with this software, and thus need well documented protocols to help guide their usage.

Currently, biological data analysis in general requires users to manually create reference points and surfaces across the image to gather information. Such a process can be lengthy and difficult when there is abundant data to parse through. As such, plugins are needed to help make the calculations

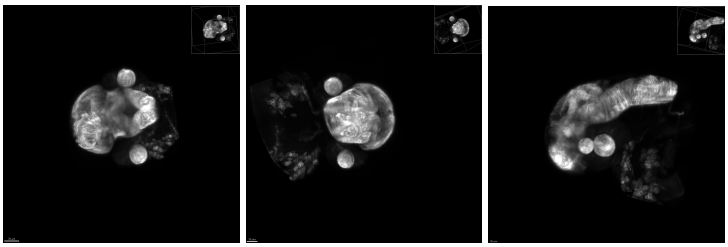

Figure 1. Zebrafish Brain

and observations easier. While many plugins exist in Imaris' official website [1], many are either for older versions of Imaris or don't interact with or calculate the statistics of the specific brain regions that interest the Barresi Lab.

Due to time constraints, we focus on two simple plugins; one that converts the data's intensity level that is above a set threshold into a new intensity and another that outputs the number of surfaces as a text file. The documentations and codes for this project are available on GitHub[2]

2. Data

Data employed in this project is in the form of z-stack imaging, which is a compilation of consecutive photographs taken at various intervals of a single zebrafish embryo. Each photograph serves as a .tif file. Together, these photographs are combined into a video, which users can view in Imaris at any specific plane of focus by freezing the video clips. Accordingly, morphological characteristics of the zebrafish embryo are also included [2]. The source of data for this study is the light-sheet microscopy system established in the Barresi Lab. More specifically, the microscopy system takes photographs of the developmental stages of a zebrafish's embryo. Files of zebrafish embryo slices then directly pipeline into the Imaris Software to create videos from 1 dimensional to 4 dimensional data files. This serves as records for each developmental trajectory (Figure 1).

Since the purpose of this project is to create workable plugins to assist the Barresi Lab in their analysis of the zebrafish embryo, specifically its brain development, the general categorization of variables in this project coincides with the nature of experimental setting. In that sense, the dependent variables are the development time for each embryonic stage, intensity of dyed embryos stratified by the factor of age, and spatial position of embryos. The independent variables are the strength of the trans-genes, tools needed in order to produce the fluorescence. What's more, the units of observation in this study is the number of embryos collected in a batch, which composes the number of fertilized eggs a female zebrafish produced in a hatch.

3. Methods

Results from previous confocal microscopy image analysis studies show that even a single package from a workable analytical program can be used for data processing.[3]. Although there exist a number of multidimensional analytical programs which allow for accurate quantification via high quality visualization (eg.BioImageXD and Icy), Imaris is the most suitable platform to use because it has quality image resolution and built-in algorithms to generate workable plugins and fundamental descriptive statistics directly in an Excel document format.

During the early planning stages of this project, deciding whether to employ ImageJ or Imaris as the major analytical program constitutes an imperative step. Since the nature of the study is to eventually generate a workable plugin to assist in fulfilling the research goals of the Barresi lab, the computable feasibility is the most essential feature to focus on when choosing an analytical

---

1 `Imaris Open`: https://imaris.oxinst.com/open/.
2 `GitHub Repository`: https://github.com/sds-capstone/barresi.

program to work with. Although both ImageJ and Imaris can generate Multi-channel, colored images, ImageJ has functions in the form of individual commands allowing customization while Imaris has pre-programmed image analysis tools organized within a clearly guided interface. One significant advantage of Imaris is its feature of allowing the user to rotate the z-stack along both the x and y-axis with 360 degrees rotation. Moreover, the utilization of 3D reconstructions in Imaris assists in visualizing sophisticated biological structures, in the case of this study, the zebrafish brain. With the capability of generating reproducible quantitative procedures embedded in Imaris for analyzing immunofluorescence labeled embryos [4], the study is able to capture more complete assessment of the neural connections between small structures of the brain, for instance, the communication activities between the two hemispheres.

We use Python (version 3.7.12) as our programming language for plugin development. For our project development purposes, Github is useful because it utilizes version control, is open source, and provides accountability. Therefore, all of the computational codes are developed using Visual Studio and uploaded to our Github repository. Then, we link Visual Studio with Imaris by adding the Imaris path and PYTHONPATH into the environment. In order to enable interaction with the Imaris API, we place `import ImarisLib.py` in the plugin file. Lastly, we group all the possible plugins into a folder and add the folder' path into Imaris so that all the plugins in that folder are easily accessible.

Prior to the actual development of our own plugin, we test accessibility in connecting ImageJ, an image processing software, to Imaris by successfully developing a "Hello World" Java plugin. This prototype plugin stored in ImageJ can be seen in the plugin console of Imaris and works by printing out the text, "Hello World", as output. We study the programming syntax and functions available in Imaris' library: *Imaris.py* embedded in the "Help" section of Imaris. Studying existing plugins also provides us with insights in structuring and laying out our own plugin. Resources that encompassed existing plugins written for Imaris include personal python XTension Github repositories (eg. Matthew's repository) and official XTension database on the website *Imaris Open*.

As the Barresi Lab is still developing their zebrafish data, they state that they would like to use Imaris to obtain some basic summary statistics manually with existing features embedded in Imaris do manual calculations. Thus, we are not fully automating their processes yet. Imaris mainly uses the data models *spots*, *surfaces*, *cells*, and *filaments* to identify and analyze all of the zebrafish data. Many of the existing plugins use *surfaces* for their calculations, but we are concerned about overlooking some important information by looking at the surface alone. Therefore we want to make our plugins also work with intensity, as the data are all set up using intensity levels initially.

During the developmental stages of our own plugin, we first focus on developing the *distance.py* plugin, which can output the farthest distance from a user selected spot to the farthest distance above a specified intensity level. More specifically, we first quantify our 3 dimensional z-stack image model through coloring the filament channels, which is an automated feature inside Imaris. The plugin can then attach a spot to the surface of the user selected cell. The average pixel intensity level is determined with the measurement of fluorescence level on the x,y,z axis. And the point is anchored at the midline on the surface of the cell, not at the center of its mass. Ultimately, the *Distance.py* plugin can return a script output in a txt.file as a result.

Cell Counter Plugin was another plugin that we developed to output the number of unique cell populations in a specific channel. After using the existing feature of Imaris to color filament channels and outlining the shapes of distinct cells, we then wrote according codes to enable the counting of the number of distinct cells which were colored in the entire model. An output suggesting the total number of cell population is printed in a popped-up window.

To debug, we first add an output section in the file. If there is an error in the plugin, there won't be an output. We then try to comment out parts where we believe it is creating the error, and then uncomment line by line to see where the problem is. We also use the output system to print out the data at various points of the code to compare differences. Once we believe we fixed the code, we run it on Visual Studio and reload Imaris. Sometimes the error is that a package isn't installed, which we

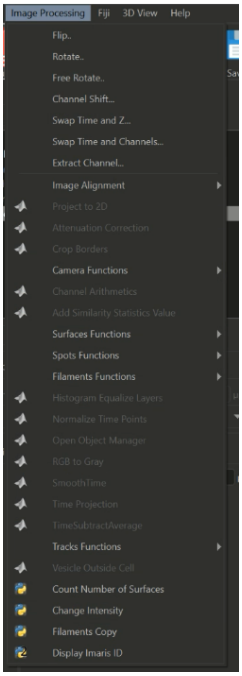

Figure 2. Plugin Location in Imaris

check by calling out the terminal of Visual Studios to check for error messages when we run the code. The error messages normally say that the package we are using is not recognized, which we solve by calling `pip install packageName` in the terminal.

4. Results

Prior to creating any plugin, we need to make sure that we have all the setup so that the users can access a working plugin. First step is to add the plugin into *Image Processing* (Figure 2) of Imaris by adding the following at the top of the plugin file:

```
#    <CustomTools>
#      <Menu>
#       <Item name="Name Shown on Menu" icon="Python3">
#         <Command>Python3XT::NameoftheFunction(%i)</Command>
#       </Item>
#      </Menu>
#    </CustomTools>
```

We can also create submenus by adding encompassing in , but choose not to as we don't have other similar plugins at the moment. We then finish the setup by linking the file to Imaris through the following code, which we use in both of our plugins. It is also a useful starter code for future plugins.

```
import ImarisLib

def NamoftheFunction(aImarisId):

  # Create an ImarisLib object
  vImarisLib = ImarisLib.ImarisLib()

  # Get an imaris object with id aImarisId
  vImarisApplication = vImarisLib.GetApplication(aImarisId)
```

```
  # Get the factory
  vFactory = vImarisApplication.GetFactory()

  # Get the currently loaded dataset
  vImage = vImarisApplication.GetDataSet()

  # Get the Surpass scene
  vSurpassScene = vImarisApplication.GetSurpassScene()
```

We create two simple plugins, one called ChangeIntensity.py and the other called CellCount.py. ChangeIntensity changes the intensity of the Z-stack image. It does this by taking in two inputs, the user's threshold intensity and intended intensity level, through the following code:

```
global Entry1, Entry2
###########################################################################
def dialog():
    global Entry1, vThreshold, Entry2, vNewHigh
    vThreshold=Entry1.get()
    vThreshold=float(vThreshold)
    vNewHigh=Entry2.get()
    vNewHigh=float(vNewHigh)
    root.destroy()

root=Tk()
root.geometry("200x50-0+0")
#Set input as the top level window
root.attributes("-topmost", True)
#########################################################################
#Set input in center on screen
# Gets the requested values of the height and width.
windowWidth = root.winfo_reqwidth()
windowHeight = root.winfo_reqheight()
# Gets both half the screen width/height and window width/height
positionRight = int(root.winfo_screenwidth()/2 - windowWidth/2)
positionDown = int(root.winfo_screenheight()/2 - windowHeight/2)
# Positions the window in the center of the page.
root.geometry("+{}+{}".format(positionRight, positionDown))

#########################################################################

Label(root,text="Targeted Intensity Level:").grid(row=0) ##grid sets the location
Entry1=Entry(root,justify='center',width=10)
Entry1.grid(row=0, column=1)
Entry1.insert(0, 100) # set default entry as 100

Label(root,text="Intended Intensity Level:").grid(row=1)
Entry2=Entry(root,justify='center',width=10)
Entry2.grid(row=1, column=1)
Entry2.insert(0, 100)
```

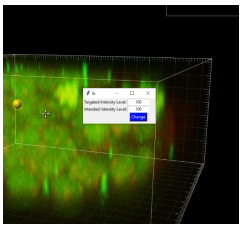

Figure 3. User Input Window

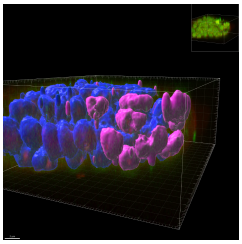

Figure 4. Adding sufaces

```
# Creates Button
Single=Button(root, text="Change Intensity", bg='blue', fg='white',command=dialog) # Previously def
# Set button location
Single.grid(row=1, column=1)
# Call the input window for display
mainloop()
```

The result of this code looks like Figure 3. It is important to note that to create the window, we need to add `import tkinter` right after we imported the ImarisLib as well.

We then change the intensity of the image using a function from ImarisLib, which changes any intensity below a certain threshold into a predefined lowest intensity level and any intensity level above the threshold into a predefined highest intensity level. Here, we set the lowest intensity level as 0, and the highest level is previously defined by user input. The channel index specifies which channel we want the change in, which we currently set as the first channel (indexed at 0). We also need to pass in the image data itself so the function knows what it is transforming.

```
vChannelIndex=0
vNewValueLow=0
vImarisApplication.GetImageProcessing().ThresholdBothChannel(vImage,vChannelIndex,vThreshold,vNewVa
```

The biggest problem with ChangeIntensity is that it directly alters the users data set, which means that once the changed data is saved there is no way to retrieve the original data. However, one of the benefits of this plugin is that it doesn't require any wrangling in the data set prior to using the plugin. On the other hand, CellCount does not directly modify the dataset and does not ask for an input. However, it does ask for the users to create a surface object first (Figure 4), of which it would count the number of surfaces. It does so through the following code:

```
vSelected = vImarisApplication.GetSurpassSelection()
vSurfaces = vFactory.ToSurfaces(vSelected)
vNumberOfSurfaces = vSurfaces.GetNumberOfSurfaces()
```

The plugin outputs the number of surfaces into a text file located in a results folder (Figure 5. The number must first be changed into a string, however, or else the code does not properly output the text file.

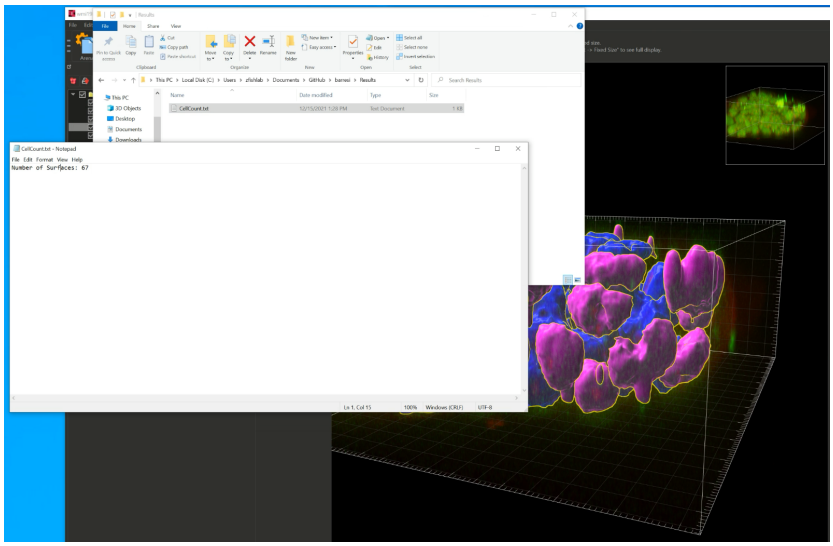

Figure 5. Output of CellCount.py

```
vText = 'Number of Surfaces: ' + str(vNumberOfSurfaces) ##stringify the number
f = open(r'C:\Users\zfishlab\Documents\GitHub\barresi\Results\CellCount.txt', 'w')
f.write(vText)
f.close()
```

4.1. Summary of Results

Table 1. Summary of ChangeIntensity Plugin

Plugin	Input	Calculations	Output
Change Intensity	Takes in the user's threshold intensity and intended intensity level	Set the Channel Index and lowest intensity value, and input all variable to ThresholdBothChannel function	N/A

Table 2. Summary of CellCount Plugin

Plugin	Input	Calculations	Output
Cell Count	N/A	Get the objects, then get the surface objects from those and count the number	Ouput number as string in text file

5. Conclusion

5.1. Limitations

Some of the limitations of this research is the speed of the light-sheet microscopy we utilize to collect data from. Since the system takes 10 seconds to take a photograph of a single clip of a Z-stack image, this limits the scope of our data extremely. Additionally, another major constraint will be its capacity in taking a given number of embryos in a given time frame, which further impedes our accessibility to the data available. For now, the capacity and the speed of the light-sheet microscopy system allows a production of 240 samples for a week if collecting a single batch a week. Mistakes in the process of data collection can also pose damage to the embryo samples.

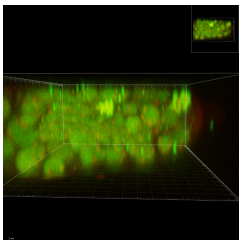

Figure 6. Frontal lobe of zebrafish

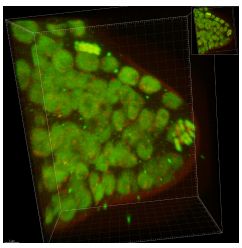

Figure 7. Frontal lobe of zebrafish

What's more, another limitation of this project is the computational speed of Imaris as an analytical program. At first, we tried to work with the real zebrafish embryo data collected from the Barresi Lab. However, constrained by the size of the data file, any computational attempts that we make code wise takes a long time to test in Imaris. With this constraint in mind, we move on to work with a smaller data file for the intention of accelerating the development of the plugins in order to have some deliverable products. Therefore, instead of using the entire zebrafish brain, we only use the data on the zebrafish frontal lobe to test the plugins (Figure 6, 7).

5.2. Ethical Statement

While the ethical concerns associated with this project are generally minimal, it is important to highlight a few factors. Firstly, we have minimal knowledge of how Imaris processes data. Our plugins could potentially permanently alter the microscopy data without us knowing about it. Additionally, as we are not biologists, we are not able to verify that the outputs of our plugins are accurate without consulting the Barresi Lab. Finally, it is crucial to make our processes reproducible for future groups by creating clear and accurate documentation. Facing the time constraints of this project, we are not sure if we disclosed all the essential information in the form of documentation in our Github repository. This possible lack of documentation can complicate the data translation process for future work based on our established framework.

5.3. Future Work & Final Thoughts

During the planning stage of this project, we outline several goals the Barresi Lab is interested in and their major goals are as followed:

a) Are bioelectric signals important in creating the early brain?
b) Does the migratorial cell population (crimneou) lay the groundwork for the neurons to form the formation of our face?
c) How do embryonic brains form for zebrafish?
d) What is the distance of each region to the farthest brain region?

Although we are not able to address these questions with accurate answers, it is beneficial for the lab's research interests that the next computational group will have enough prior information in developing their own plugins or continue to develop the plugin that we provide the initial code for.

For the next group to continue our work, we advise that the best way to start is by understanding the Imaris software first by studying the tutorials offered on the Imaris official website. Another useful

resource would be to get in contact with the Bitplane tech team when running into technological obstacles within Imaris. They can give live tutorials and answer specific questions about Imaris.

Within our Github readme file, we encapsulate some key information pertaining to setting up Imaris for creating python plugins and how to start coding a plugin. With this information about setting up and establishing connections between various computing software, we hope that the next computational group working with Barresi Lab will be able to start creating plugins that are more tailored for the lab's needs. The next group can reference the open source plugins and our own plugins to create new ones that are more complex and specifically made to work with zebrafish data.

A major challenge for our plugin creation was understanding the functions provided by the ImarisLib python library. Since having proper function documentation would have helped us with navigating and utilizing functions, we believe that the next group can create personalized documentation on documentation on the ImarisLib library, which can be shared as an open source. When learning about ImarisLib functions, we recommend specifically looking at function parameters and return type values. An easy logic error in a plugin would be trying to take the return value of one function and inputting it as the parameter of another only to realize that the data types do not match.

Although we were unable to finish the *distance plugin*, we are satisfied by what we are able to accomplish in such a short time. This is especially the case given the limited and confusing documentation. The future computational group can take a look at our *distance plugin* as referenced in the **Methods Section**. The next group can use this to get familiar with structuring and laying out a future plugin. Part of the code for this plugin is in our Github repository, which can be the starter code for the next group. We also outline the potential output in the **Methods Section**, which can be a useful reference for the future groups.

Acknowledgments: For the completion of this work, we would like to acknowledge the contribution of Micheal Barresi, head of the Barresi Lab, for sponsoring and providing necessary guidance for this project; Professor Ben Baumer for his guidance throughout the project; The Beckman Foundation for providing the grant to acquire the advanced light-sheet microscope; Matthew Gastinger, Advanced Application Scientist at Bitplane technology company, for his support in navigating Imaris and for the access to his open source Imaris plugins; and the students on the Barresi Lab's computational and forebrain teams.

Author Contributions: The Abstract, introduction, and acknowledgement are written by all four authors.
Data is written by Summer.
The Methods Section is written by Summer with an additional paragraph from Catherine.
The Results Section is written by Catherine.
The Conclusion is written by Carol, Haley, and Summer. Limitations by Summer, Ethical Statement by Haley, Future Work and Final Thoughts by Carol.
The whole paper is extensively edited by all four authors.
Catherine and Carol connected the paths and PYTHONPATH for Imaris plugin creation.
Catherine and Summer mainly worked on the plugins. Catherine commented on the code.
Carol and Haley completed the Github readme documentation.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gautier, M.K.; Ginsberg, S.D. A method for quantification of vesicular compartments within cells using 3D reconstructed confocal z-stacks: Comparison of ImageJ and Imaris to count early endosomes within basal forebrain cholinergic neurons. *Journal of Neuroscience Methods* **2021**, *350*, 109038. doi:https://doi.org/10.1016/j.jneumeth.2020.109038.
2. Light sheet fluorescence microscopy.
3. Vadim Zinchuk, O.G.Z. **2009**. *44*, 125–172. doi:https://doi.org/10.1016/j.proghi.2009.03.001.
4. Miura, K.; Tosi, S.; Möhl, C.; Zhang, C.; Paul-Gilloteaux, P.; Schulze, U.; Norrelykke, S.F.; Tischer, C.; Pengo, T. Bioimage Analysis Tools. In *Bioimage Data Analysis*; Miura, K., Ed.; Wiley-VCH, 2016.