**AutoPipeline Manual**

API for Batch Processing DREAM.3D

**Tyler Weihing (SOCHE), Dr. Joseph C. Tucker (Exponent)**

**July 08, 2016**

Contents

[Introduction 3](#_Toc455736560)

[Getting Started with AutoPipeline 3](#_Toc455736561)

[Download 3](#_Toc455736562)

[Installation and Setup 3](#_Toc455736563)

[Support 3](#_Toc455736564)

[AutoPipeline Tutorials 4](#_Toc455736565)

[Setting Up a Test Sample 4](#_Toc455736566)

[Multiple Identical Runs 4](#_Toc455736567)

[Iterating Input Parameters in Pipelines 6](#_Toc455736568)

[A Few Key Notes 9](#_Toc455736569)

# Introduction

The AutoPipeline is a GUI program that allows for batch processing of DREAM.3D pipelines (.json files for DREAM.3D Version 6). DREAM.3D in it of itself does not support iterative processing of pipelines, and thus the user incurs a large time burden when required to run multiple instances of DREAM.3D. With AutoPipeline, the user is able to load in a predetermined pipeline file, specify the properties to iterate, and batch process the group.

1. Software Requirements
   1. DREAM.3D
   2. Paraview (for visualization)
   3. Windows
   4. MATLAB Runtime Library (for deployed executable, if not building from source)
2. Terms of Use: Please refer to DREAM.3D License [here](http://dream3d.bluequartz.net/?page_id=448).

# Getting Started with AutoPipeline

The following sections provide the necessary information for new users to obtain and set up AutoPipeline. This involves downloading the folder from the DREAM.3D website. Developers or those interested in maintaining an up-to-date version of the code should consider pulling or cloning a copy from the git repository [here](https://github.com/dream3d/DREAM3DMatlab).

## Download

The AutoPipeline package is available at <<this website>>. The developers will strive to keep the AutoPipeline version compatible with the current stable release of DREAM.3D.

## Installation and Setup

Once the AutoPipeline folder is finished downloading, unzip and move the folder to the location of your choice. Double-click the .mcr file and the program should begin to install. During the installation, you will be prompted about the MATLAB runtime. Make sure to download this and place it in a folder of your choice.

**IMPORTANT:** When the program is finished downloading, move the two text files (“input\_filters\_list.txt” and “write\_filters\_list.txt”) into the folder containing the AutoPipeline executable. If this is not done properly, the program will not work!

## Support

AutoPipeline is not commercial code and there are no guarantees or claims, stated or implied, pertaining to its fitness for any purpose. For all questions and inquiries, please contact Joe Tucker (jtucker@exponent.com).

# AutoPipeline Tutorials

The following sections present a few short tutorials on running AutoPipeline. It is important that the user have a fully functioning DREAM.3D pipeline before attempting to use this software. The user will benefit from having [Paraview](http://www.paraview.org/) installed on their computer for verification of the process. Refer to the Paraview website and documentation for any questions regarding this program.

## Setting Up a Test Sample

For the tutorials in this manual, you will need to create a synthetic volume within DREAM.3D. To do this, open DREAM3D and load the “(1) Single Cubic Phase Equiaxed” prebuilt pipeline in the “Workshop” folder. Set the input file in the Read DREAM.3D File filter to CubicSingleEquiaxed.dream3d located in DREAM3D/Data.

In the Initialize Synthetic Volume filter, set the dimensions to 100x100x100, and the resolutions to 1. Name the file “ap\_test.dream3d” in the last filter, check the “Write Xdmf” box and place this in a work folder of your choosing. Click run and once it is finished you can make sure the synthetic structure was created properly via Paraview.

## Multiple Identical Runs

Some situations call for multiple runs of an identical pipeline. Depending on how many times the pipeline needs to be run and how long a pipeline takes to complete, the use of AutoPipeline can make this more efficient. In this sample case, creating more than one synthetic structure with the same stats file is demonstrated.

As mentioned earlier, it is important to have a working DREAM.3D pipeline in order to utilize AutoPipeline. To create one for this demonstration, use the pipeline from the example above. With the pipeline open, select File -> Save As and save the pipeline as “synthetic\_builder.json” in your work folder.

Once you have the saved pipeline, open the AutoPipeline program. When the program is first opened, the screen should look like this:

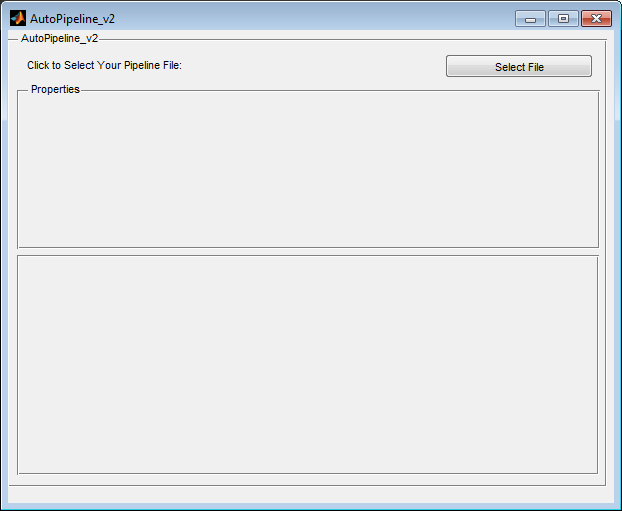


Figure : AutoPipeline Start Screen

Click the “Select File” button and browse for your pipeline file you just created and select “Open.” You will now see a window titled “Filter Names,” as well as “Property Names.” These will be important in the upcoming tutorials. For our purposes, next to the window titled “Select Inputs” is a Select button. Press this and a new window should appear that looks like the following:

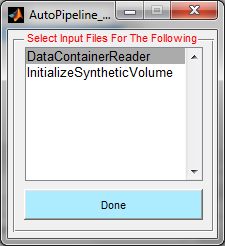


Figure : Input File Window

The options in the list box are the names of the filters that have the requirement for some type of input. “DataContainerReader” refers to the ReadDream3dFile filter and “InitializeSyntheticVolume” refers to the filter of the same name. For this tutorial, the input files for both filters are the same. First, click the DataContainerReader selection. Select the file from DREAM3D\Data titled “CubicSingleEquiaxed.dream3d” you used to create your test sample. Repeat this for the InitializeSyntheticVolume selection. Once this is completed, click “Done” on this window to return to the main program screen.

Next, click the Select button next to “Select Output Names and Folder.” A new window should open in which you can select which write filters you want to save. In this example, the “Write Dream3d Data File” filter is the only option in the list. It should be highlighted by default. In the “Enter Save Name” box type in the name you wish the files you are creating to be called. In the example, we will again use “ap\_test.” In this GUI, the file suffix is not required. Click “Confirm and Select Folder” and browse to the folder you’d like to save the new files in and click “Select Folder.”

Once the outputs window closes, you will be back at the main screen. Now click the Select button next to “Select Pipeline Runner Executable.” Browse to your DREAM.3D folder and at the base level there should be an application called “PipelineRunner.exe.” This is the command line version of DREAM.3D that AutoPipeline uses to run DREAM.3D. Select this application and click Open and the window will close. Your main screen should now look similar to this:

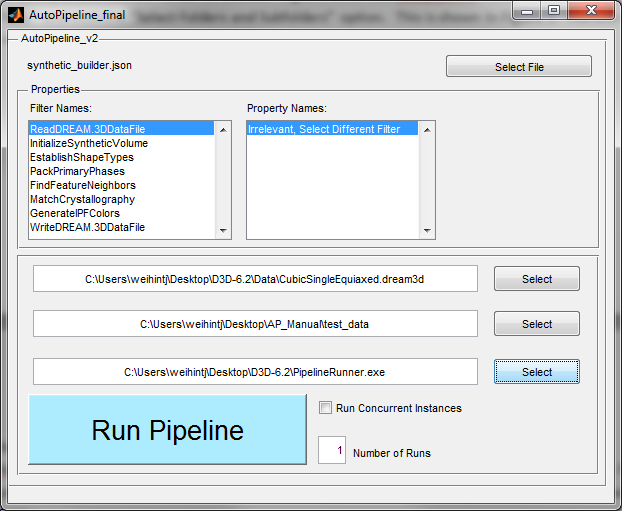


Figure : AutoPipeline during Workflow

For this example, we are focusing on the field next to “Number of Runs.” The purpose of the number in this box, default is set to 1, is to provide an overall number of times you want to run the exact same scheme. Set this number equal to 3. Having 3 in this box will mean that your pipeline “synthetic\_builder,” will run a total of 3 times with the same values. In this case, you will end up with 3 synthetic structures all of which are 100x100x100 with a resolution of 1. However, when the structures are created, DREAM.3D randomly samples from the statistics file you’ve provided; therefore, you will have 3 similar boxes in size/shape, but the grain structure within will vary. This means you now have an easy way of creating multiple statistically similar synthetic structures with one click!

Click on Run Pipeline and the program will save the files to the output folder you selected. Once the program is finished running, you should see both a .dream3d file as well as an .xdmf file for each structure you created for a total of 6. The structures can be verified in Paraview with the .xdmf files.

You will probably notice the files have a unique naming system. In addition to the save name that was entered in the output GUI, there is a run number and iter number. The run number is related to the Number of Runs value you changed in this section. This is to help keep track of the files created by this program. The iter number is related to the iterative parameter changing that is covered in the next section.

## Iterating Input Parameters in Pipelines

In this next tutorial, a demonstration will be given in AutoPipeline’s ability to iterate DREAM.3D parameters. This means that if you have one or more parameters you’d like to adjust in your pipeline, i.e. a tolerance, AutoPipeline can do the work for you.

For this example, create a pipeline in DREAM.3D, which includes the following filters: Read DREAM.3D Data File, Crop Geometry, and Write DREAM.3D Data File. In the “Read DREAM.3D Data File” filter, select ap\_test.dream3d that you created earlier in this guide.

In the “Crop Geometry” filter, make sure X-, Y-, and Z-Mins are set to zero. X-Max and Y-Max should be set to 99, and Z-Max should be zero. Uncheck the “Renumber Features,” “Update Origin,” and “Save As” New Data Container” boxes. The Cell Attribute Matrix pulldown menus should be SyntheticVolumeDataContainer -> CellData.

Finally, the “Write DREAM.3D” filter should have an output file name set to your working folder.

Save this pipeline in the same manner as the previous example and open AutoPipeline. Click “Select File” and choose the pipeline you just saved. Next, in the “Filter Names” window, select “Crop Geometry.” You should notice the Min, Max, and Step windows appear on your screen. Your main screen should now look like this:

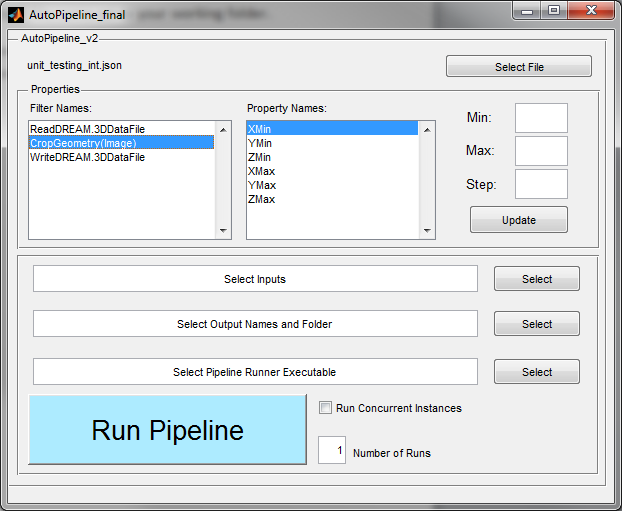


Figure : Min, Max, Step Windows

This is arguably the strongest feature in AutoPipeline. Whenever a filter with a parameter AutoPipeline can iterate is selected, these windows will populate the main screen. Here you can select an array of values over which you would like AutoPipeline to iterate for a parameter. Additionally, when more than one parameter is edited within the GUI, the program will run every permutation of the changes entered in.

In the case of our tutorial, we will be cropping our synthetic structure along a varying z-axis. This means we do not need to change X or Y-axis values because we set them to the correct values in the pipeline itself. The only value we need to change is the Z-Max value. For our purposes, we will change the value of the Z-Axis from 0 to 2. This means the first file created will be a single voxel thick slice of the synthetic cube we created earlier. The second file will be 2 voxels thick, and the third will be 3 voxels thick. For this purpose, we want the Z‑Max to start at 0, end at 2, and have a step size of 1.

In order to enter the values, select the Min field and enter zero. Then select Max and enter 2. Finally, enter 1 in the Step field. Once the values are entered, click the “Update” button under the text boxes. **IMPORTANT:** If the “Update” button is not clicked, the values entered will not be saved and the increment will not occur.

Once this is complete, enter the input (our ap\_test.dream3d from earlier), output, and pipeline executable in the same manner as the last example. When Run Pipeline is clicked, 3 files will be created; each a slice of our original structure. Checking these files in Paraview will validate our expected results.

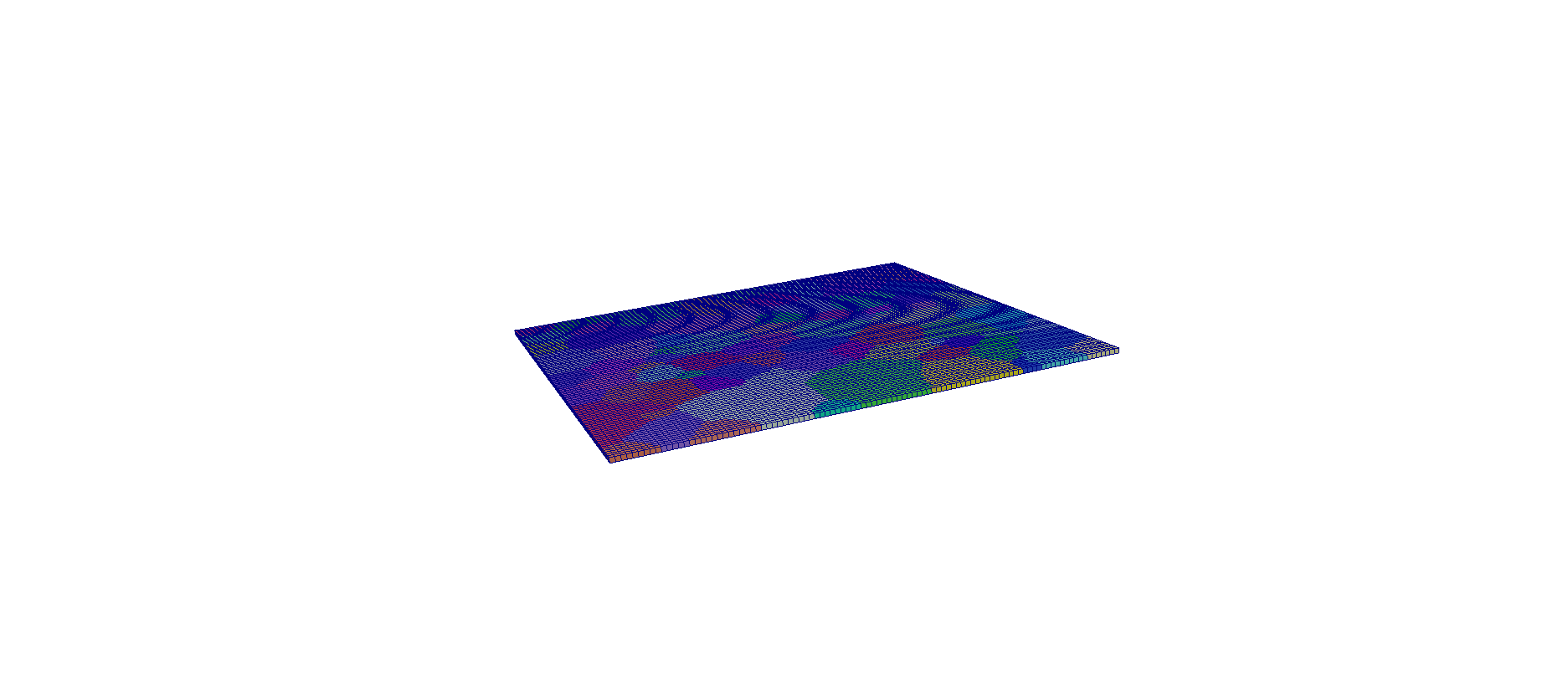


Figure : Single Voxel Thick Slice

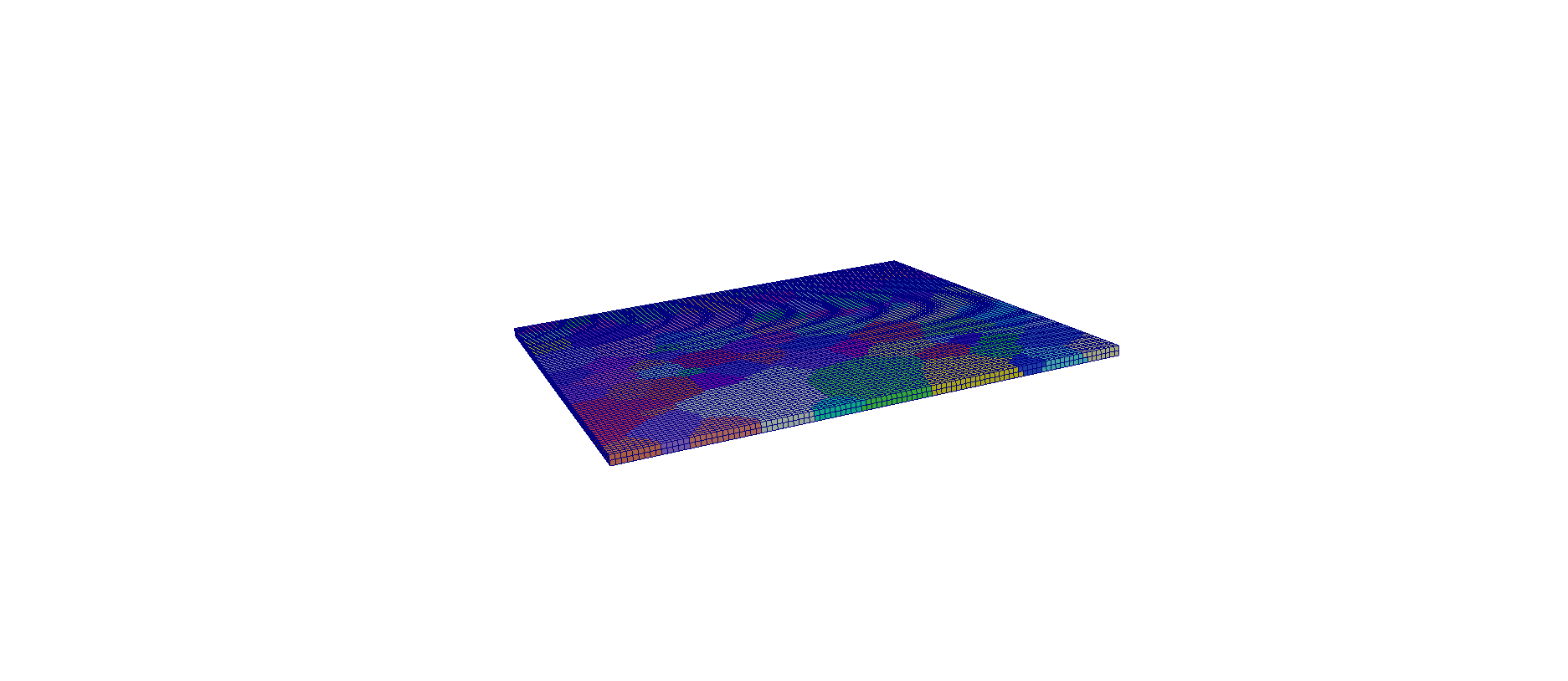


Figure : Two Voxel Thick Slice

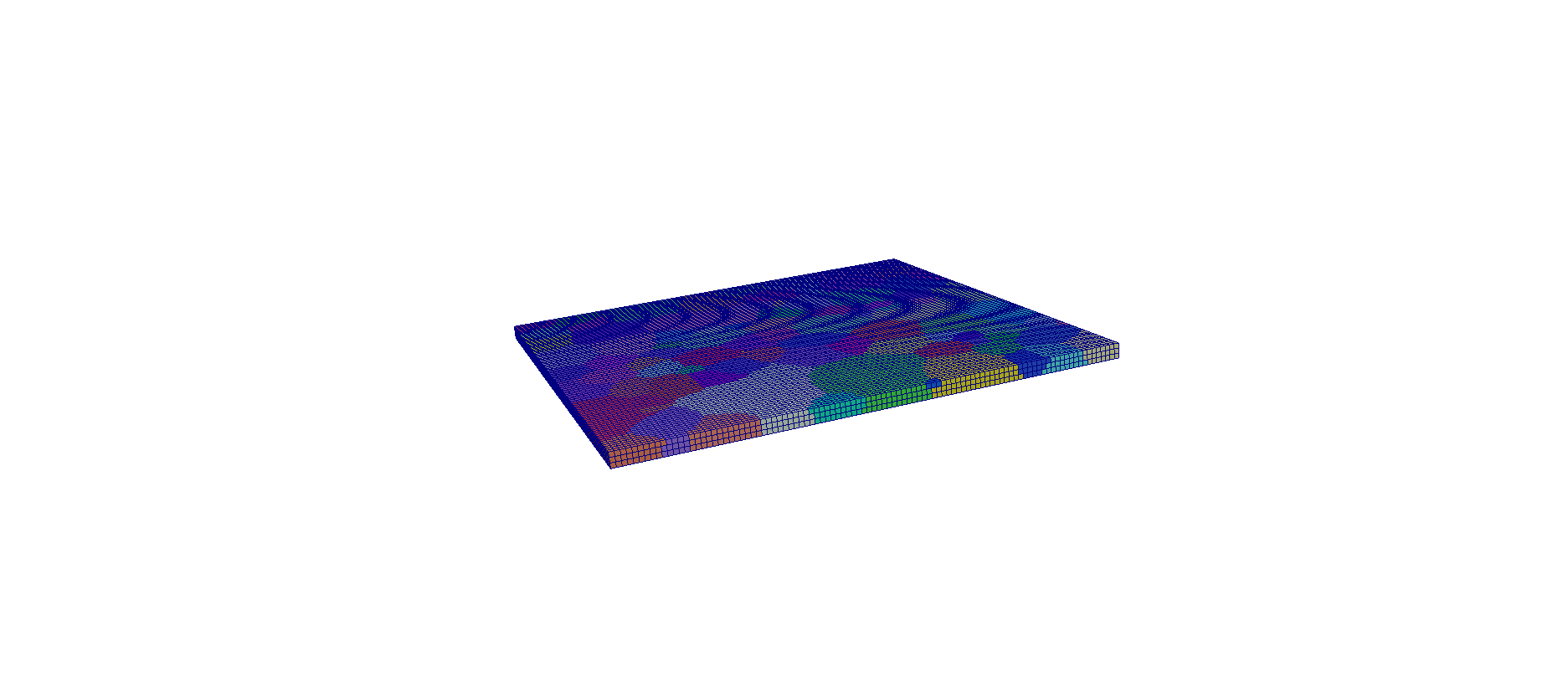


Figure : Three Voxel Thick Slice

You can see the z-axis is incrementing by one; therefore, each slice is growing thicker. Success! There is also an excel file in the save location. This is a library that is created each time the program’s iterative ability is used. Inside, you will find the name of each file created, and what the settings for each parameter are in that specific file.

# A Few Key Notes

1. AutoPipeline’s interation ability only works on objects considered an “integer” or “int” values and “doubles.” These include almost all changeable values in DREAM3D that are not considered a “vec3.” A vec3 is a vector of 3 numbers, and is typically found when a direction is required, i.e. and IPF figure.
2. An Input file MUST be entered each time the program is used. If this does not happen the program will not work. Multiple inputs may also be used. Treat this as an extra layer to the iterative process. For example, if you are changing 2 parameters, have Number of Runs set to 2, and have 2 inputs, It will iterate the 2 parameters twice for each input.
3. When using the incrementing functionality of AutoPipeline, runtime can get quite large. The developers suggest you try a single run of the pipeline you create and gauge its runtime. Total time to run all iterations of a multi-property pipeline will be equal to the product of the run time and the number of values for each property, e.g., if the run time is 30 seconds and you vary z-max by 5 values, and z-min by 8 values, the total run time would be (30 seconds)x(5 values)x(8 values) = **20 mins**. Keep this in mind when running many pipelines consecutively, it can get quite lengthy!