

# Using DREAM3D to construct an initial microstructure for PRISMS-CPFE

Aaditya Lakshmanan<sup>1</sup> and Mohammadreza Yaghoobi<sup>2</sup>

<sup>1</sup>Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA

<sup>2</sup>Department of Materials Science and Engineering, University of Michigan, MI 48109, USA

## Introduction

This tutorial is a guide towards synthetic microstructure reconstruction using [DREAM.3D](#)<sup>®</sup>, in a format compatible with PRISMS-CPFE. PRISMS-CPFE uses two files - (i) **grainID.txt** and (ii) **orientations.txt**. **grainID.txt** contains the grainIDs for every voxel arranged in a specific order based on the convention chosen by DREAM.3D. The file **orientations.txt** contains the orientation information, including every distinct grain ID and the corresponding orientation represented as a Rodrigues vector.

## Procedure

The package on which the present procedure has been tested, is DREAM.3D Version 6.4.197. As a reminder, Dream.3D incorporates the passive<sup>1</sup> convention of rotations and this convention must be ensured in the experimental data provided.

1. Browse to the directory **DREAM3D-6.4.197-Win64/DREAM3D-6.4.197-Win64** and double-click on the DREAM3D executable. This should bring up two windows i) The DREAM3D toolbox containing all the filters required to construct a pipeline and (ii) The pipeline window where filters can be assembled to create a pipeline. Filters will be copied from the toolbox to the pipeline window sequentially, after which the pipeline will be started to generate the required files. For details pertaining to specific filters refer to the [DREAM3D manual](#).
2. Copy the **StatsGenerator** filter to the pipeline window. This filter is useful to generate the statistics based on which the microstructure is reconstructed. The statistics referred to here is the grain-size distribution, ODF(Orientation Distribution Function), shape distribution and MDF(Misorientation Distribution Function). These can be specified either through the parameters window or an external file. Before we start specifying this microstructural information, we need to specify something about the material itself. This can be done by clicking on the yellow gear button on the top-left corner after which a window pops up to input phase properties. Select crystal structure as **Hexagonal**, **Fraction** as **1**(we are dealing with a single phase polycrystal) and **Phase Name** as 'Magnesium', as shown in Fig. 1
3. Ensure that the tab displaying **Size distribution** is currently active or it can be made so by just clicking on it. The grain size distribution should be selected in a way that it can mimic that of the real specimen. In the present case, the selected values are - Mu = 4, Sigma = 0.5, Bin Step Size = 6.0000, Min Cut Off = 2 , Max Cut Off = 1.5( Fig. 2). This will result in an estimated 16 feature bins. For

---

<sup>1</sup>Passive convention here refers to the passive transformation, in which orientation descriptors are specified in a way to transform the global coordinate system to a local coordinate system. In the present case it is the transformation from the sample reference frame to the crystal reference frame.

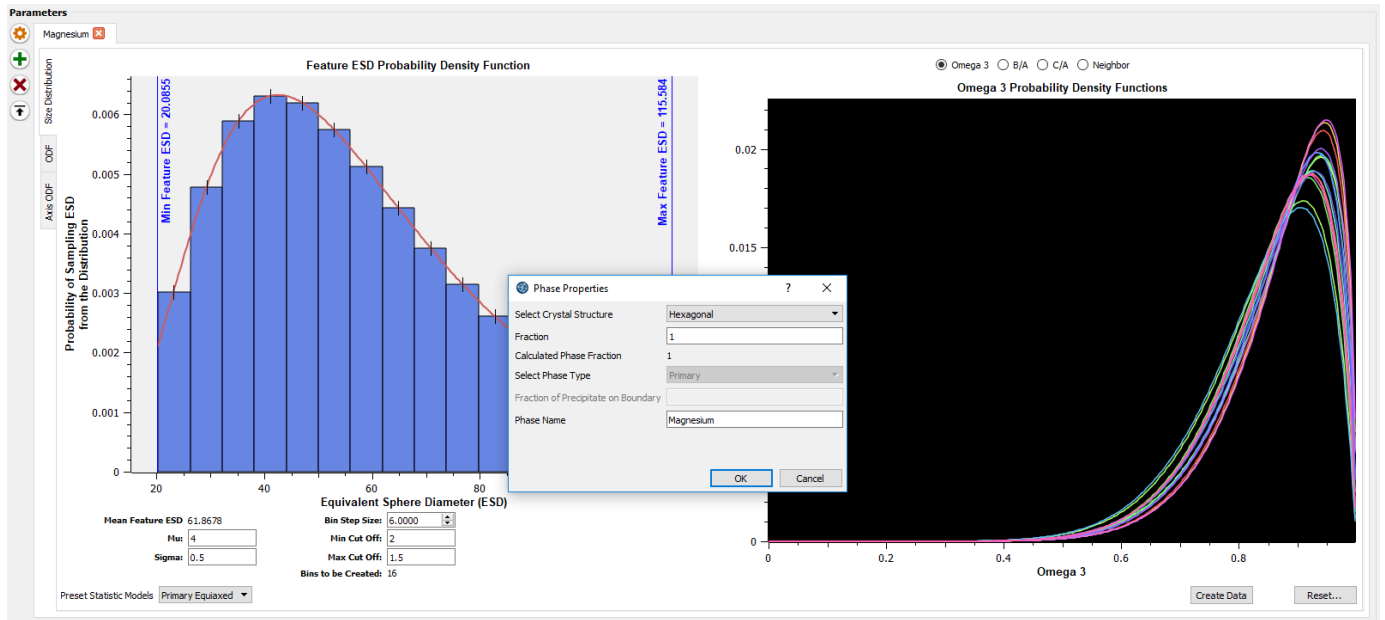


Figure 1: Size distribution

the **Preset Statistic models** choose **Primary Equiaxed** because we will be working with equiaxed grains. Then click on the **Create Data** button to update the statistics for the new set of parameters.

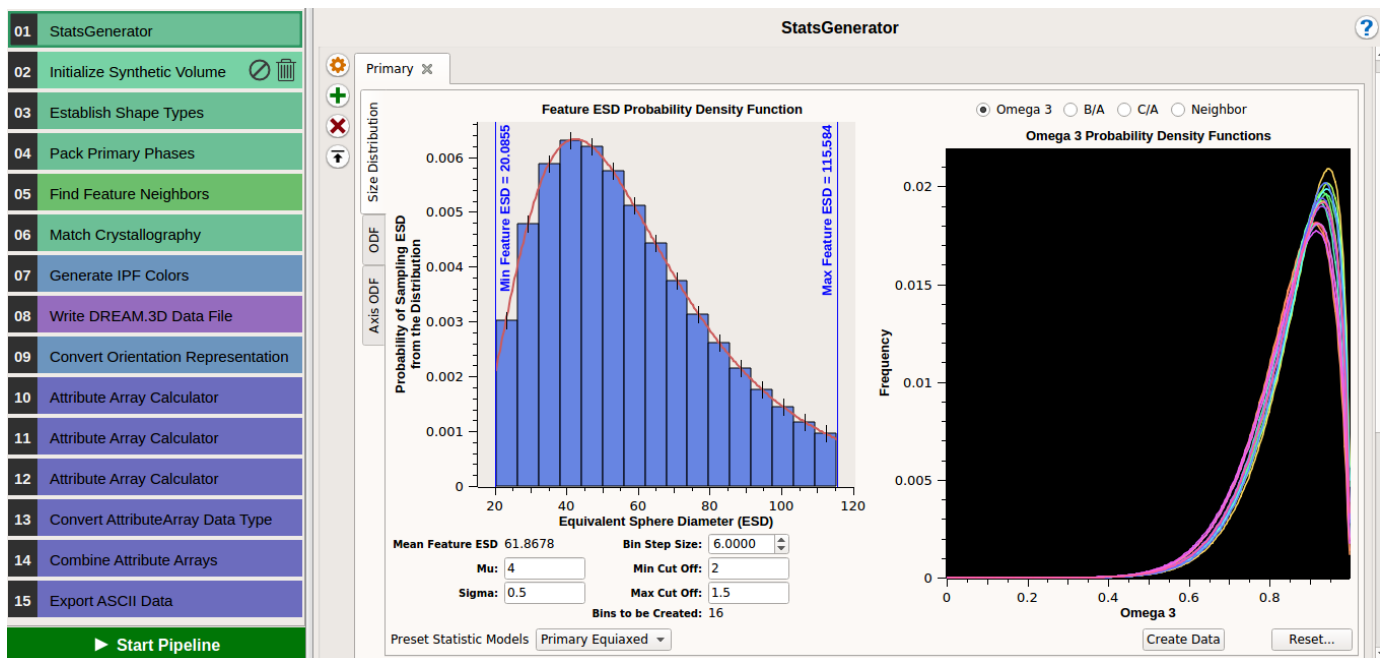


Figure 2: Size distribution

- Notice that the 2 tabs displaying **ODF** and **Axis ODF** are accessible. Click on the **ODF** button and click on the radiobox under **OFD parameters** displaying **Bulk Load From File**. The first textbox asks for the input file which can be added by clicking on the **Select** button and choosing the required file. In the present case, the Euler angles are specified in radians hence, leave the box next to **Values**

in **Degrees** unchecked. Choose the angle representation as **Euler** and the delimiter as **Space**, as in Fig. 3. Then click on the button displaying **Load Data** to generate the pole figures.

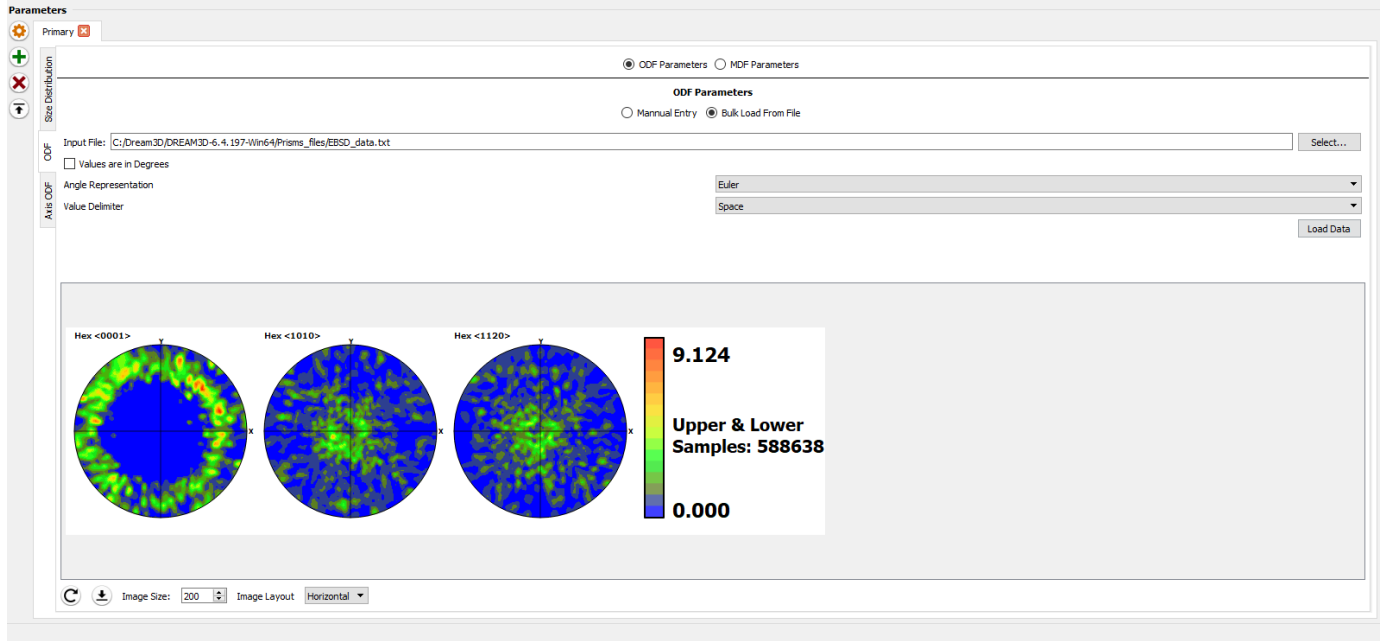


Figure 3: Reading orientation data from an external file

Alternately, one can explicitly specify the texture data by specifying the Euler angles, the intensity and spread via the set the parameters as follows - Euler 1 = 0 , Euler 2 = 0 , Euler 3 = 0 , Weight = 50000, Sigma = 14(Fig. 4). Pressing the keyboard **Enter** key will update the ODF and generate the corresponding pole figures. Crystallographic texture is controlled using these parameters. The 2D EBSD data should be used to generate the sample texture.

5. Click on the tab displaying **Axis ODF** and set the parameters as follows - Euler 1 = 0, Euler 2 = 0 , Euler 3 = 0 , Weight = 150 , Sigma = 3 (Fig. 5) - and press the keyboard **Enter** key to update the axis ODF. Morphological texture is controlled using these parameters.
6. Copy the **Initialize Synthetic Volume** filter to the pipeline window where the box dimensions can be specified. The box is split into voxels according to the dimensions as follows - if the dimension value in the first textbox is 50, then the box will be sectioned into 50 partitions along direction '1' to produce 50 voxels in that direction. Similarly the second and third textbox refer to the dimensions along directions '2' and '3' respectively. The resolution refers to the length of the box along each direction, so that the larger the resolution, the more grains generated for a given size. By ticking the checkbox next to '**Estimate number of features**' one can check the estimated number of grains that would be generated for the chosen dimensions and resolution. In the present case, we choose a dimension of 10 in each direction with a resolution value of 10(Fig. 6) . Note that the number of features estimated can change on multiple tests because the entire processing is essentially that of sampling based on a probability distribution.
7. Copy the **Establish Shape Types** filter to the pipeline window. Here one can choose the approximate shape of the grain. The available options are ellipsoid, superellipsoid, cube octahedron and cylinder(A,B,C). We choose the grain shape as an ellipsoid (Fig. 7).
8. Copy the **Pack Primary Phases** filter to the pipeline window. This filter will place primary Features with the sizes, shapes, physical orientations and locations corresponding to the goal statistics. Additionally, periodic boundary conditions are enforced by checking the box next to **Periodic Boundaries**(Fig. 8), because in most cases an RVE is to be modelled.

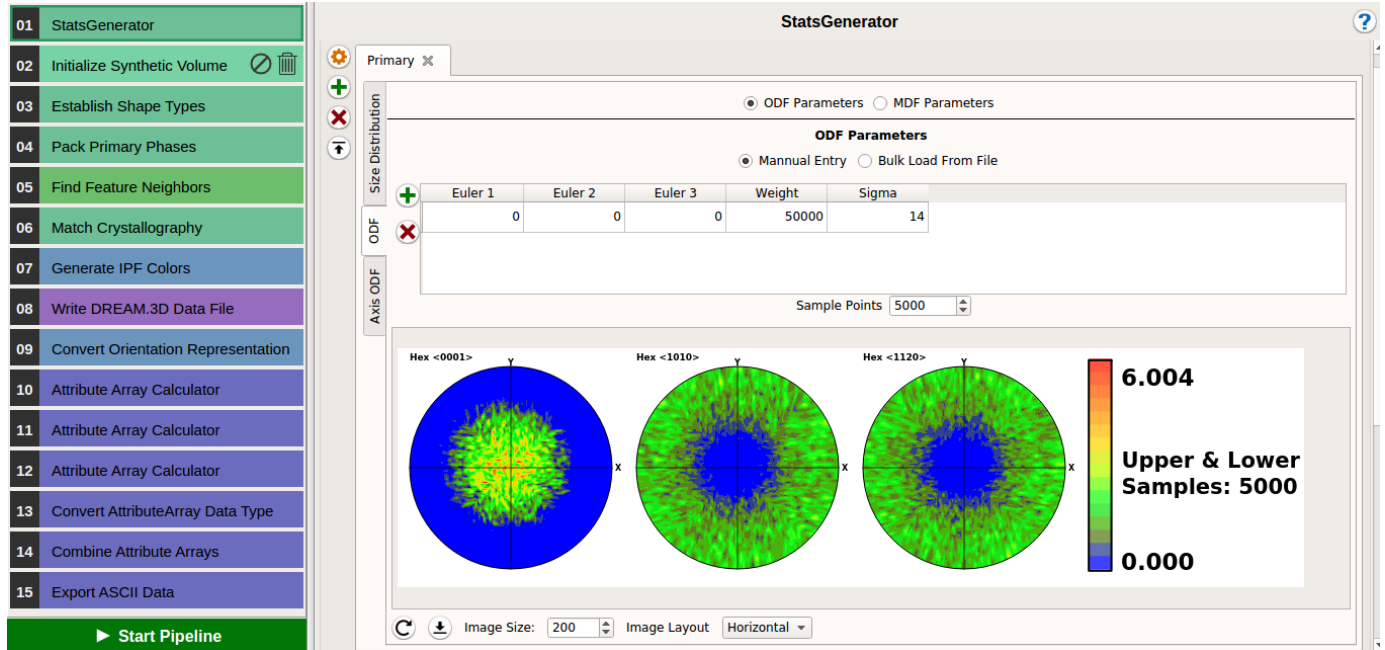


Figure 4: ODF

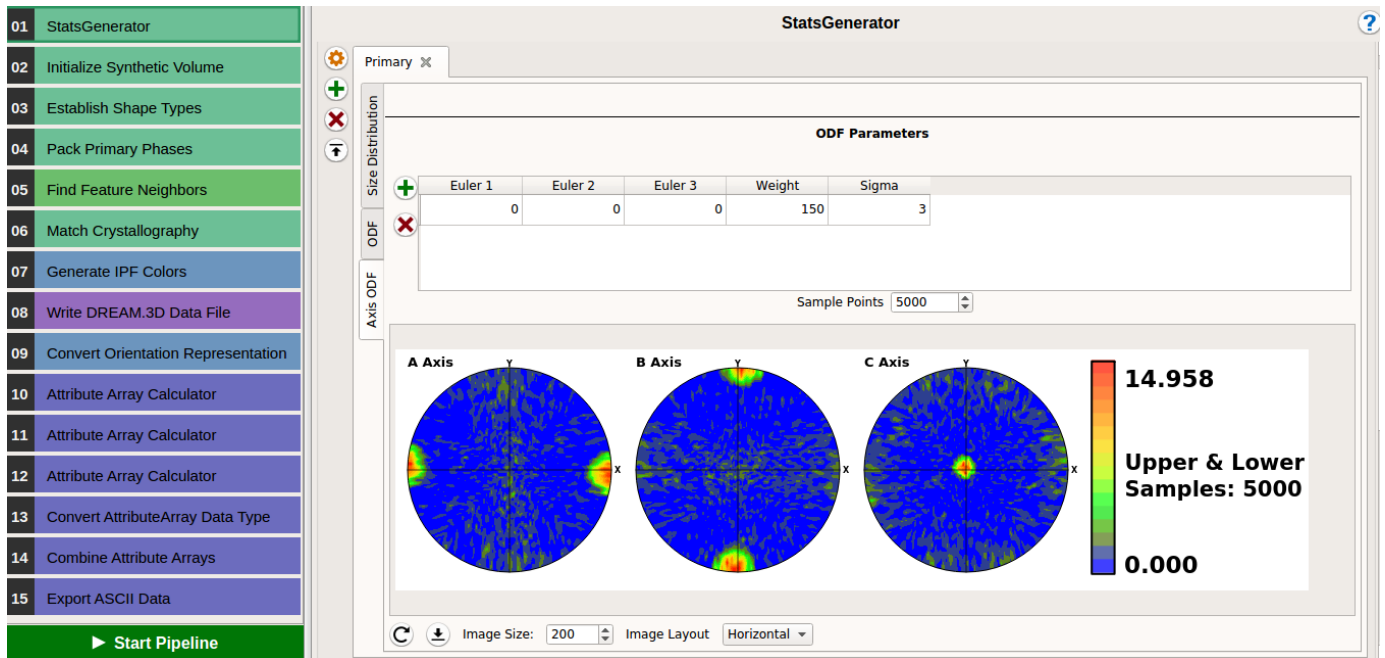


Figure 5: Axis ODF

9. Copy the **Find Feature Neighbors** filter to the pipeline window(Fig. 9). This filter looks for features neighboring any particular feature, and this action is performed over all generated features.
10. Copy the **Match Crystallography** filter to the pipeline window, which attempts to match a predefined orientation distribution function (ODF) to a set of Features(Fig. 10). Once the pipeline is started, this step is the most time-consuming process because a number of iterations are performed until a satisfactory match is established between the supplied statistics and the those synthetically generated.

**Initialize Synthetic Volume**

**Parameters**

☐ Estimate Number of Features

Dimensions: 10 10 10

Resolution: 10 10 10

Origin: 0 0 0

**Required Objects**

Cell Ensemble Data: StatsGeneratorDataContainer / CellEnsembleData / Statistics

**Created Objects**

Synthetic Volume Data Container: SyntheticVolumeDataContainer

Cell Data: CellData

Ensemble Attribute Matrix: CellEnsembleData

**Start Pipeline**

Figure 6: Initialize synthetic volume

**Establish Shape Types**

**Required Objects**

Phase Types: StatsGeneratorDataContainer / CellEnsembleData / PhaseTypes

**Created Objects**

Shape Types: ShapeTypes

Primary: Ellipsoid

**Start Pipeline**

Figure 7: Establish shape types

11. Copy the **Generate IPF Colors** filter to the pipeline window to render colors to every grain depending on the orientation representation in the stereographic triangle. Set the reference direction to [ 0 0 1](Fig. 11).
12. Copy the **Export Dx File (Feature Ids)** filter to the pipeline window. This filter generates an

**Pack Primary Phases**

**Parameters**

☒ Periodic Boundaries

☐ Use Mask

☐ Already Have Features

Save Shape Description Arrays: Do Not Save

**Required Objects**

**Cell Data**

Cell Attribute Matrix: SyntheticVolumeDataContainer / CellData

**Cell Ensemble Data**

Statistics: StatsGeneratorDataContainer / CellEnsembleData / Statistics

Phase Types: StatsGeneratorDataContainer / CellEnsembleData / PhaseTypes

Phase Names: StatsGeneratorDataContainer / CellEnsembleData / PhaseName

Shape Types: StatsGeneratorDataContainer / CellEnsembleData / ShapeTypes

**Created Objects**

**Cell Data**

Feature Ids: FeatureIds

Phases: Phases

**Cell Feature Data**

Cell Feature Attribute Matrix: CellFeatureData

Figure 8: Pack primary phases

**Find Feature Neighbors**

**Parameters**

☐ Store Boundary Cells Array

☒ Store Surface Features Array

**Required Objects**

**Cell Data**

Feature Ids: SyntheticVolumeDataContainer / CellData / FeatureIds

**Cell Feature Data**

Cell Feature Attribute Matrix: SyntheticVolumeDataContainer / CellFeatureData

**Created Objects**

**Cell Data**

**Cell Feature Data**

Number of Neighbors: NumNeighbors

Neighbor List: NeighborList

Shared Surface Area List: SharedSurfaceAreaList

Surface Features: SurfaceFeatures

Figure 9: Find feature neighbors

output file containing the grain-ids for all the voxels in the format of **grainID.txt**. In the present case, the output file is specified with the name **grainID.txt** as in Fig. 12. This file includes the information

**Match Crystallography**

**Parameters**

Maximum Number of Iterations (Swaps)

**Required Objects**

**Cell Data**

Feature Ids

**Cell Feature Data**

Phases

Surface Features

Neighbor List

Shared Surface Area List

**Cell Ensemble Data**

Statistics

Crystal Structures

Phase Types

Number of Features

**Created Objects**

**Cell Data**

Euler Angles

**Cell Feature Data**

Figure 10: Match crystallography

**Generate IPF Colors**

**Parameters**

Reference Direction

☐ Apply to Good Elements Only (Bad Elements Will Be Black)

**Required Objects**

**Element Data**

Euler Angles

Phases

**Ensemble Data**

Crystal Structures

**Created Objects**

**Element Data**

IPF Colors

Figure 11: Generate IPF colors

for  $N_1 \times N_2 \times N_3$  voxels where  $N_1$ ,  $N_2$ , and  $N_3$  are the values selected for **Dimensions** in **Initialize Synthetic Volume** (Fig. 12). The numbering convention is included in [Appendix B](#).

13. Copy the **Write DREAM3D Data File** filter to the pipeline window (Fig. 13). This is performed so that the microstructure can be visualized in a visualization software like [ParaView](#) (Fig. 14).
14. We now focus on generating the input file **orientations.txt**. PRISMS-CPFE reads the Rodrigues vec-

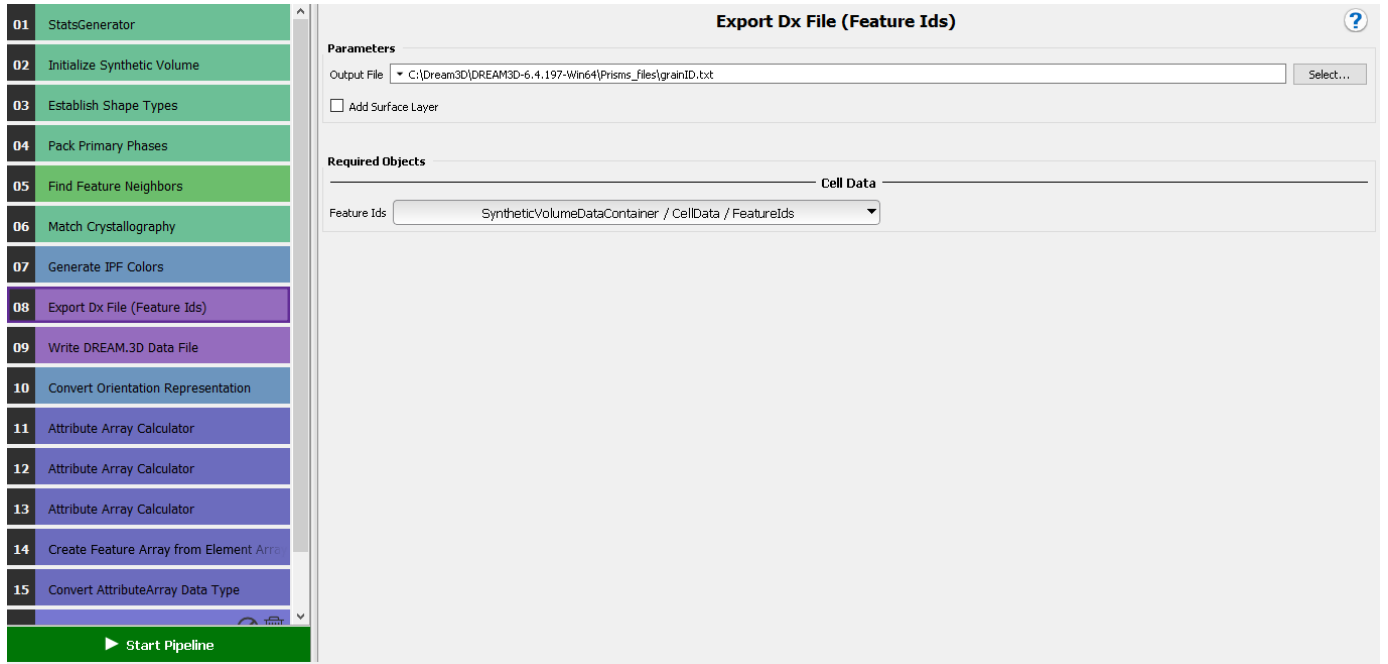


Figure 12: Export grainID.txt

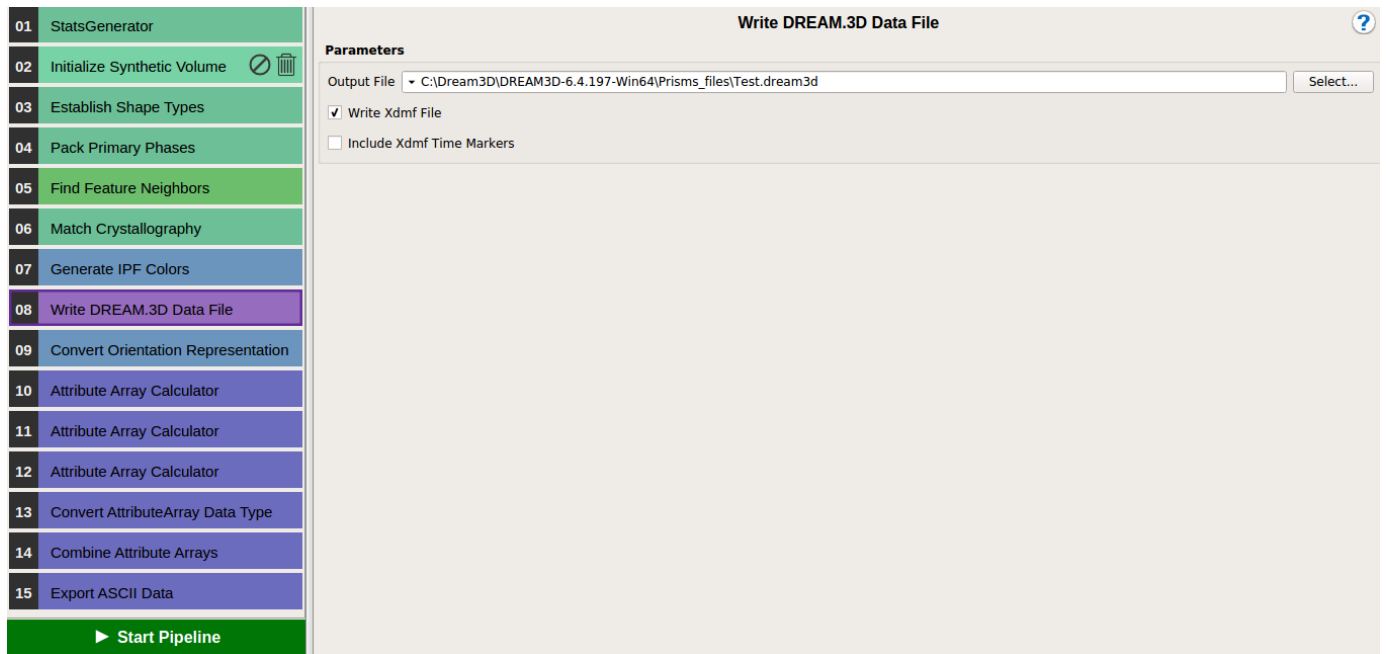


Figure 13: Write DREAM.3D data file

tor for the **orientations.txt** input file. Accordingly, some post-processing steps should be performed on the existing data containers to convert orientation information from Euler angles(Bunge) to the corresponding Rodrigues vector. To do this, copy the **Convert Orientation Representation** filter to the pipeline window. Then choose the input orientation type as **Euler** and the output orientation type as **Rodrigues**. For the **Input Orientations** options choose **SyntheticVolumeDataContainer / CellFeatureData / EulerAngles**, and prescribe a name for the output orientation. Here we choose



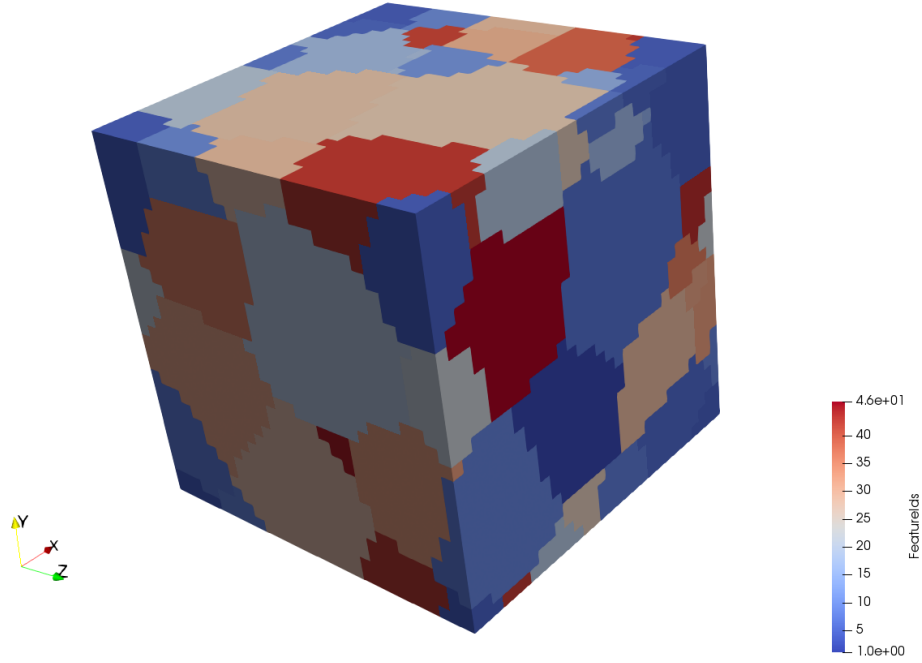


Figure 14: Sample microstructure visualized in ParaView

the name **rvec**. We then obtain a 4-component vector in which the first 3 components are identified with the axis of rotation (a unit vector) and the 4<sup>th</sup> component is  $\tan(\frac{\theta}{2})$  where  $\theta$  is the angle of rotation taken to be positive when anti-clockwise while staring into the vector. The output data container is named **rvec**(Fig. 15 ).

15. In the previous step the Rodrigues representation of the Euler angles was obtained, which is a 4-component vector and not in its most concise form. PRISMS-CPFE reads the ‘true’ Rodrigues vector which can be obtained from a 4 component representation by multiplying the first three values (unit vector components) by the magnitude (the 4<sup>th</sup> component). DREAM3D currently does not permit creating a vector of a smaller size from the given vector, hence we use a workaround. We separate out each individual component (which DREAM3D allows), perform some mathematical operations and recombine them into the desired 3-component vector. The following steps can be pursued in that direction.

- Copy the **Attribute Array Calculator** filter to the pipeline window. Set **Cell Attribute Matrix** to **SyntheticVolumeContainer / CellFeatureData**. In the textbox type **-rvec[0]\*rvec[3]**, which means that we are multiplying the 1<sup>st</sup> component of **rvec** with the 4<sup>th</sup> component to generate the first component of the ‘actual’ vector. Under **Created Objects** set the option of **Calculated Array** to **SyntheticVolumeContainer / CellFeatureData** and the value as **rvec.comp1**(Fig. 16).
- Copy the **Attribute Array Calculator** filter to the pipeline window. Set **Cell Attribute Matrix** to **SyntheticVolumeContainer / CellFeatureData**. In the textbox type **-rvec[1]\*rvec[3]**, which means that we are multiplying the 2<sup>nd</sup> component of **rvec** with the 4<sup>th</sup> component to generate the first component of the ‘actual’ vector. Under **Created Objects** set the option of **Calculated Array** to **SyntheticVolumeContainer / CellFeatureData** and the value as **rvec.comp2**.
- Copy the **Attribute Array Calculator** filter to the pipeline window. Set **Cell Attribute Ma-**

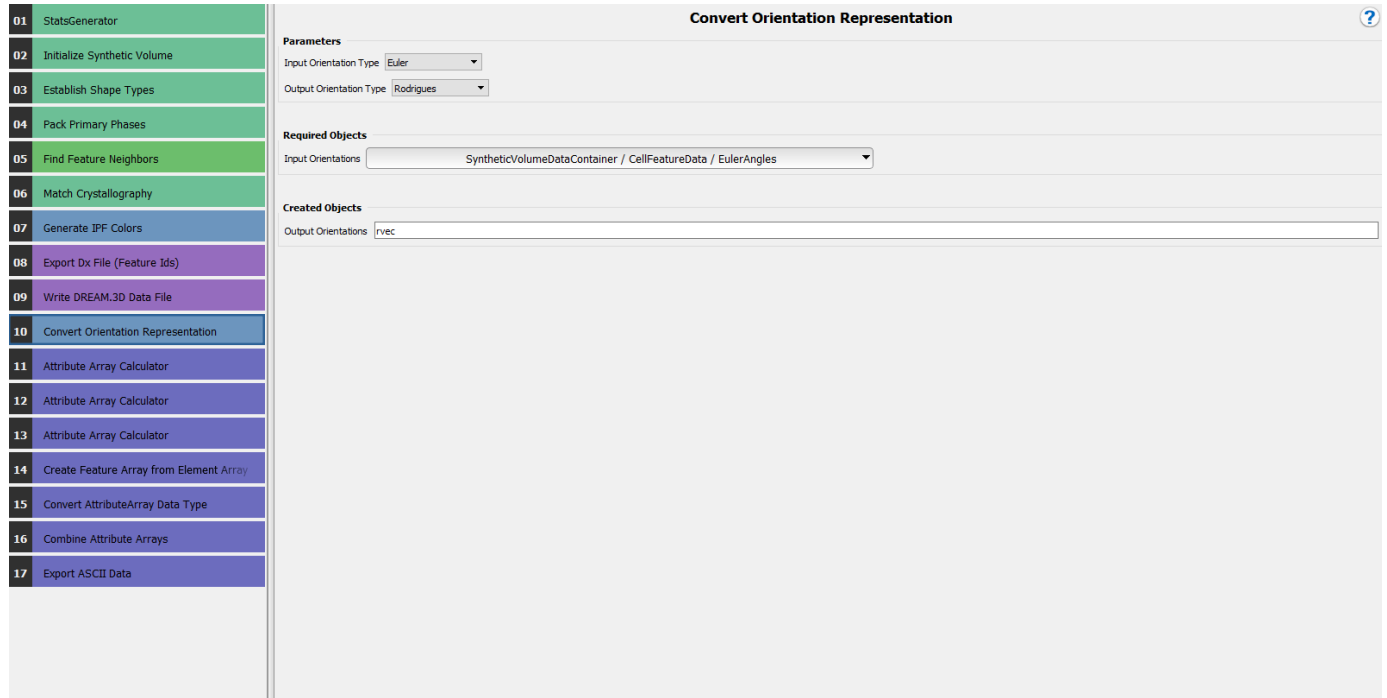


Figure 15: Convert orientation representation

**trix** to **SyntheticVolumeContainer / CellFeatureData**. In the textbox type **-rvec[2]\*rvec[3]**, which means that we are multiplying the 3<sup>rd</sup> component of **rvec** with the 4<sup>th</sup> component to generate the first component of the 'actual' vector. Under **Created Objects** set the option of **Calculated Array** to **SyntheticVolumeContainer / CellFeatureData** and the value as **rvec.comp3**.

16. From the previous procedure we have the Rodrigues vector. What we are left to obtain is the feature ids. For this copy the **Create Feature Array from Element Array** filter to the pipeline window. What we are doing here is to convert the feature ids available for every voxel into an array of distinct feature ids for the entire microstructure. In the **Required Objects** tab choose **SyntheticVolumeContainer / CellData / FeatureIds** for both the options. In the **Created Objects** tab choose **SyntheticVolumeContainer / CellFeatureData** for the **Feature Attribute Matrix** option and name the **Copied Attribute Array** as **featids**( Fig. 17 ) .
17. We now wish to combine the feature ids and the Rodrigues vector into a single data container so that we can write it to an output file. However, the two data types don't currently match because the feature ids are 32 bit integers while the Rodrigues vectors are of type double. A conversion to data type double needs to be performed. To accomplish this, copy the **Convert AttributeArray Data Type** filter to the pipeline window. For **Scalar Type** choose **Double 64 bit**, for **Attribute Array to Convert** choose **SyntheticVolumeContainer / CellFeatureData / FeatureIds**, and name the **Converted Array Attribute** as **featids\_conv**( Fig. 18 ). Any name can be specified though.
18. We are now ready to combine the required data into a single data container. Copy the **Combine Attribute Arrays** filter to the pipeline window. For **Attribute Arrays to Combine** choose **SyntheticVolumeDataContainer / CellFeatureData**. Then from the **Available Data Arrays** double-click on the names **featids\_conv**, **rvec.comp1**, **rvec.comp2** and **rvec.comp3**. Name the new data container as **orientations**, beside **Combined Data** (Fig. 19). We are now ready to export this data to an ASCII file so that the PRISMS-CPFE code can read it in.
19. Copy the **Export ASCII Data** filter to the pipeline window. Specify the output path as required,

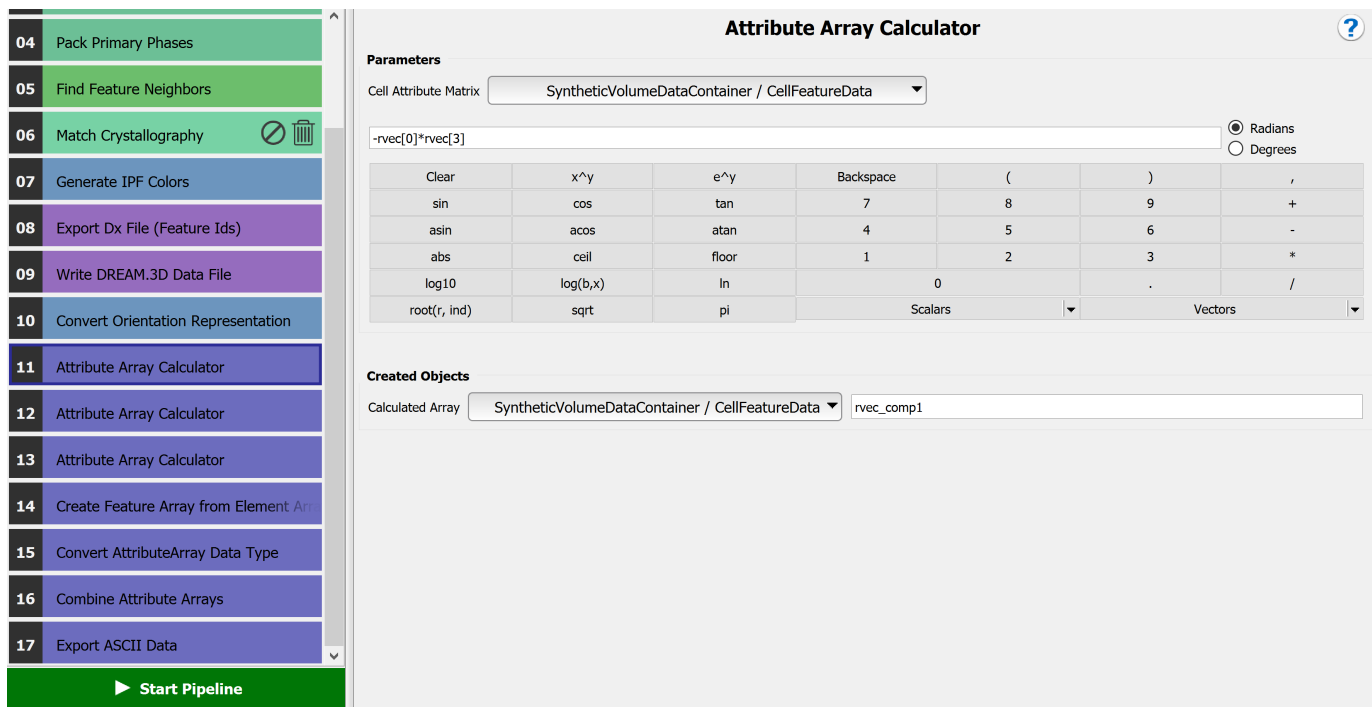


Figure 16: Attribute array calculator

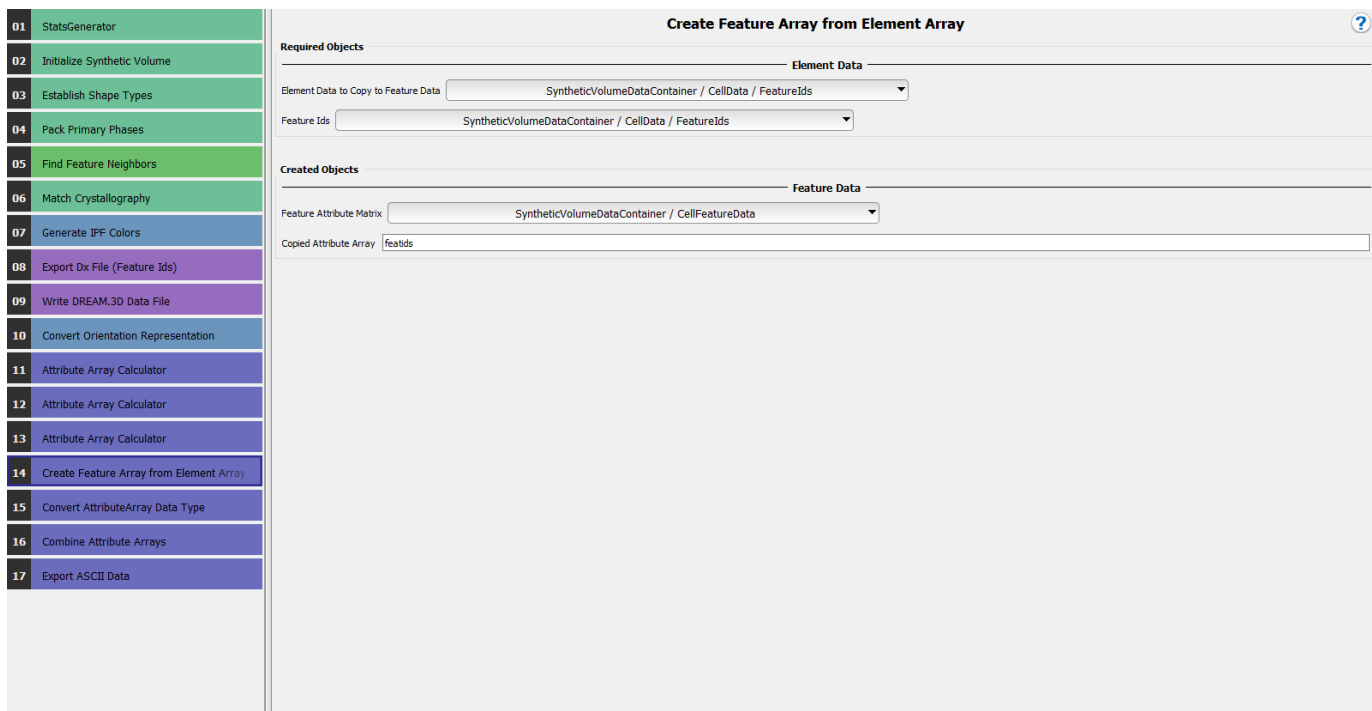


Figure 17: Attribute array calculator

file extension as **.txt**, maximum tuples per line as 1 and **(space)** as the delimiter. For **Attribute Arrays to Export** choose **SyntheticVolumeDataContainer / CellData**, and then double-click on the data container **orientations**( Fig. 20). This will eventually generate the file **orientations.txt**.

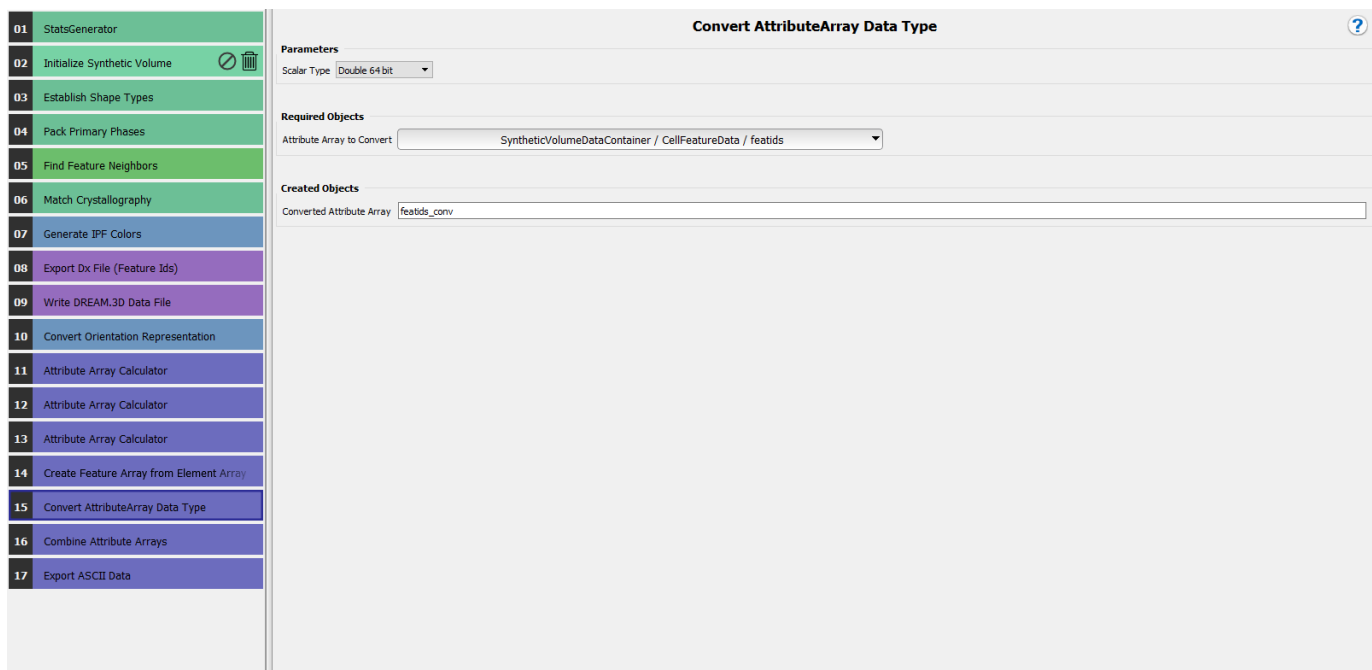


Figure 18: Convert attribute array data type

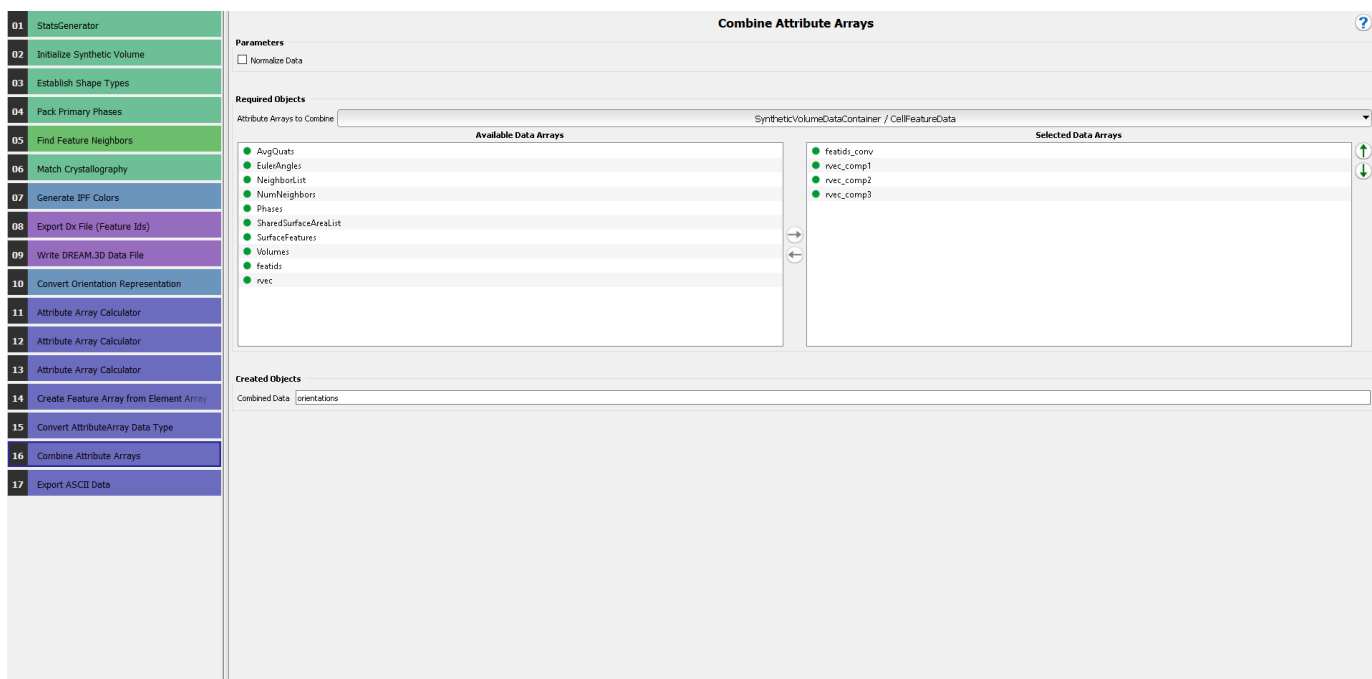


Figure 19: Combine attribute arrays

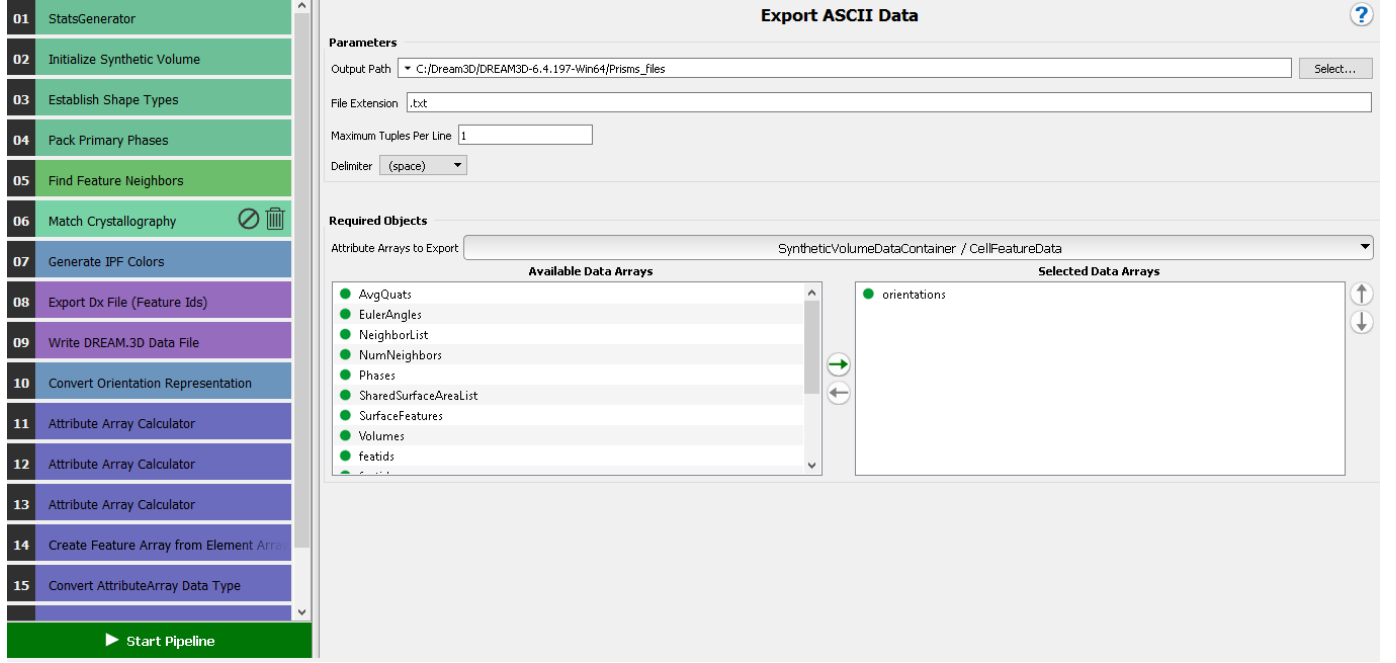


Figure 20: Export orientations.txt

We have concluded the creation of the pipeline and are ready to generate the orientations file. Click on **Start Pipeline** and after some wait, the file **grainID.txt** and **orientations.txt** will be generated.

## Appendix A - Numbering convention in grainID.txt

Referring to the filter **Initialize Synthetic Volume**, let  $N_1$ ,  $N_2$ , and  $N_3$  be the number of voxels in directions 1, 2 and 3 respectively. Then **grainID.txt** contains grain IDs for all the voxels specified in an arrangement consisting of  $N_3$  columns and  $N_1 \times N_2$  rows. Referring to Fig. 14, there is a coordinate system associated with the microstructure aligned with the box edges. These are labelled as x, y and z. Every voxel can be uniquely identified by its position with respect to this coordinate system and for the structured grid, integer descriptors will suffice. For example, the corner voxel closest to the origin may be indexed as (1, 1, 1) because it is in the first voxel in x, y and z directions respectively. Using this notation, any voxel can be indexed as  $(i, j, k)$  with  $i \in \{1, 2, \dots, N_1\}$ ,  $j \in \{1, 2, \dots, N_2\}$ ,  $k \in \{1, 2, \dots, N_3\}$ . Then the grain ID of such a voxel will be present in the  $[j + (i - 1) * N_2]^{th}$  row and in the  $k^{th}$  column of **grainID.txt**. A snapshot of the file is included in Fig. 21.

```

1  # object 1 are the regular positions. The grid is 10 10 10. The origin is
2  # at [0 0 0], and the deltas are 1 in the first and third dimensions, and
3  # 2 in the second dimension
4  #
5  object 1 class gridpositions counts 10 10 10
6  origin 0 0 0
7  delta 1 0 0
8  delta 0 1 0
9  delta 0 0 1
10 #
11 # object 2 are the regular connections
12 #
13 object 2 class gridconnections counts 10 10 10
14 #
15 # object 3 are the data, which are in a one-to-one correspondence with
16 # the positions ("dep" on positions). The positions increment in the order
17 # "last index varies fastest", i.e. (x0, y0, z0), (x0, y0, z1), (x0, y0, z2),
18 # (x0, y1, z0), etc.
19 #
20 object 3 class array type int rank 0 items 1000 data follows
21 42 42 42 49 49 49 49 49 49 35
22 42 42 39 39 9 9 9 9 35 35
23 39 39 39 39 39 9 9 45 45 35
24 52 39 39 39 39 39 45 45 45 45
25 52 8 39 53 53 53 53 36 36 36
26 8 46 53 53 53 41 41 36 36 36
27 33 46 53 53 53 53 53 24 36 36
28 33 33 33 53 43 43 43 24 24 10
29 42 33 42 53 49 49 49 49 49 10
30 42 42 42 49 49 49 49 49 49 42
31 42 42 42 42 9 9 9 9 25 35
32 35 42 39 39 9 9 9 9 35 35
33 35 39 39 39 39 9 9 9 35 35
34 52 39 39 18 18 38 45 45 45 12
35 52 46 39 18 18 41 41 45 52 12

```

Figure 21: Snapshot - **grainIDs.txt**