EBSDPolygonizer User Manual

Dr Jun Liu

Version 1.0

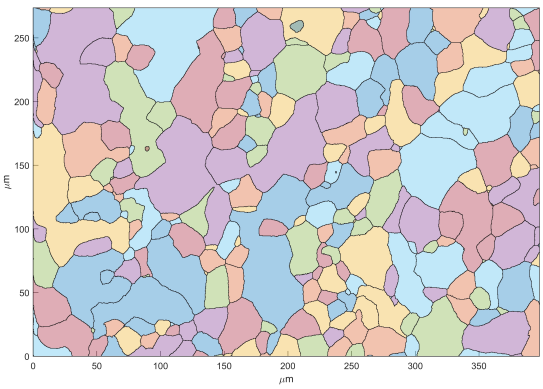
October 23, 2023

Chapter 1

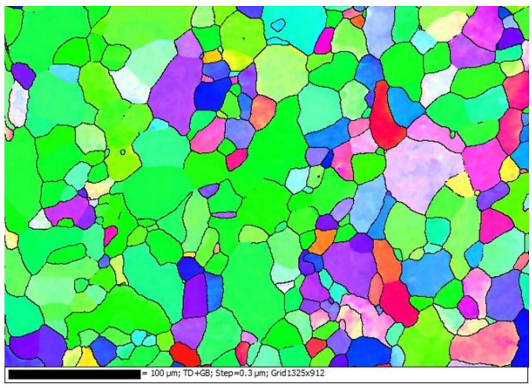
MATLAB APP with GUI

1.1 Introduction

`EBSDPolygonizer` is a MATLAB-based app designed to transform EBSD grain data into polygonal representations. In this conversion, each grain is represented by a distinct polygon. This transformation allows for the direct utilization of real microstructures as geometries in finite element (FE) models and automated assignment of material parameters and boundary conditions to individual grains. Figure 1.1 illustrates a representative example, contrasting the converted microstructure with its corresponding inverse pole figure (IPF) map.



(a) Polygonized microstructure



(b) IPF Map

Figure 1.1: Typical example of converted microstructure (a) compared to the corresponding IPF map (b).

;

1.2 Installation

Download and run the MATLAB installer file `.mlappinstall`. Alternatively, click on the icon 

in the toolbar under the APPS ribbon (see Figure 1.2), navigate to the installation file and install.

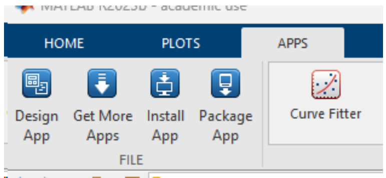


Figure 1.2: Install App from the MATLAB window.

The installed App will appear in the APPS drop-down panel (see Figure 1.3).

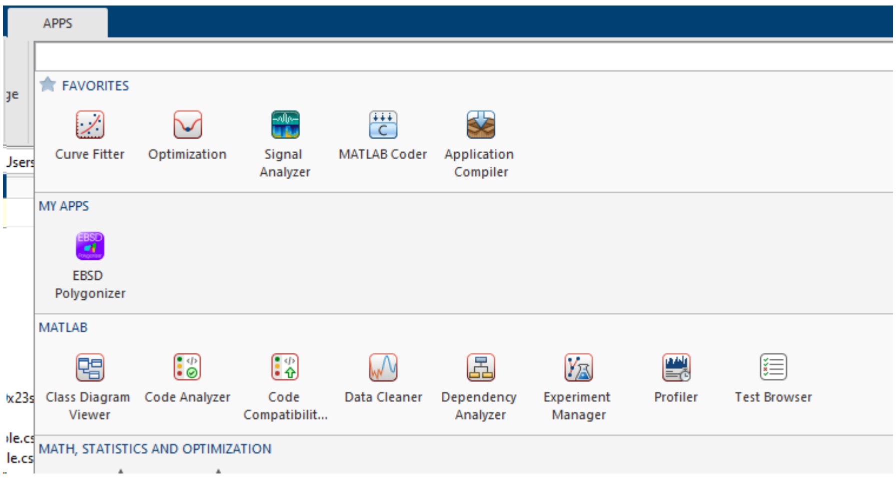


Figure 1.3: Installed EBSD Polygonizer in the APPS list.

1.3 Tutorial

1.3.1 Pre-process and export EBSD data

`EBSDPolygonizer` currently supports single-phase EBSD grain data. Essential inputs include two text files (pixel file and grain file) and the EBSD scanning step size. Details such as pixel information and grain IDs are contained within these files. The pixel file contains details for each EBSD pixels and the grain ID to which it belongs. The grain file contains the grain ID and the mean crystallographic orientation represented by three Euler angles. The former is usually much larger size than the latter. These two files can be exported from `AZtecCrystal` or `HKL Channel 5`.

Pre-processing

1. Remove all non-indexed pixels using cleaning functions available in software packages like AZtecCrystal or HKL Channel 5.
2. Utilize the Grain Detection function to identify grains within the dataset.

Prepare Pixel and Grain Files

- **HKL Channel 5**
  1. Right-click within the Grain Detection window
  2. Select export grain list to export the pixel file.
  3. Select export all cells to export grain file
- **AZtecCrystal**
  1. Detailed instruction to be added.
- **Customised `.csv` file**
  1. **pixel file:** `.csv` file containing four columns of data with a header row being the column names including `Index`, `X`, `Y`, `GrainID`.
  2. **grain file:** `.csv` file containing four columns of grain data with a header row being the column names including `Id`, `Mean Orientation phi1`, `Mean Orientation PHI` and `Mean Orientation phi2`.

1.3.2 Import pixel and grain data and step size

1. Start the EBSD Polygonizer app.
2. Define the EBSD data type. For this tutorial, click on the `HKL Channel 5`.
3. Input step size or export from a `*.cpr` file. Once imported or input, the `step size given` indicator turns green.

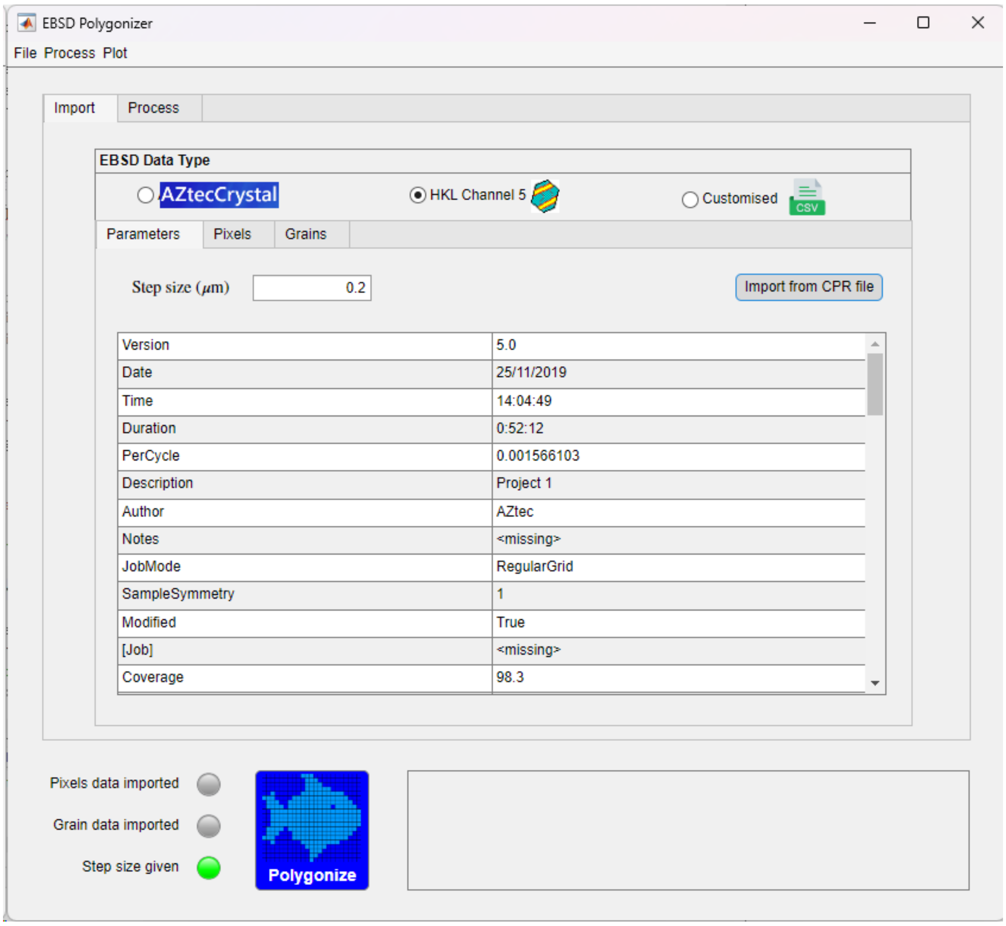


Figure 1.4: Import step size and EBSD information from `*.cpr` file

4. Go to `Pixels` tab.

5. Click on `Select pixels data` button to select the pixel file and import. Alternatively, input the file path in the text box next to the button. Click on `Preview and Import` button to import the pixel data. Make sure the Pixels data imported indicator turns green. See Figure 1.5.

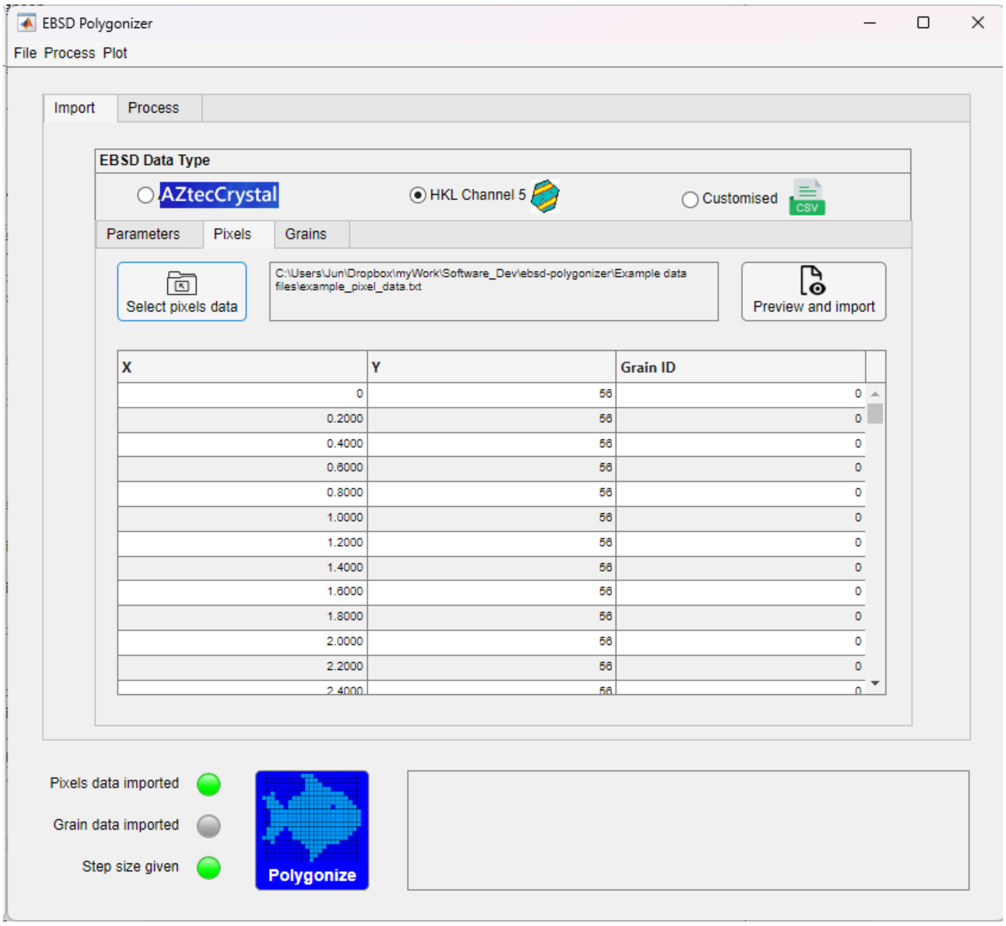


Figure 1.5: Import pixel data

6. Go to `Grains` tab
7. Select grain file to import ensuring the `Grain data imported` indicator turned green (See Figure 1.6).

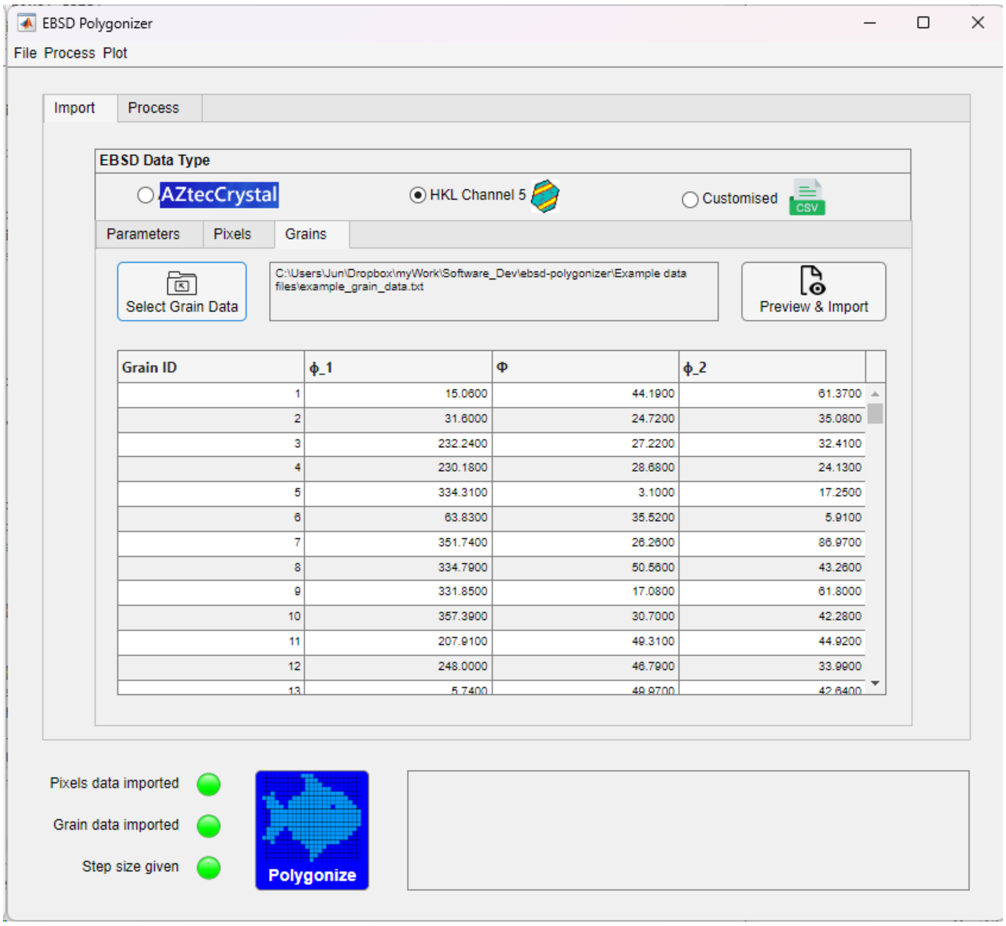


Figure 1.6: Import grain data file

1.3.3 Polygonize

1. Once all required data is imported and verified, click on the Polygonize button to initiate the polygonization process.

2. The application will provide progress updates, and upon completion, display the results for review and further processing. See Figure 1.7

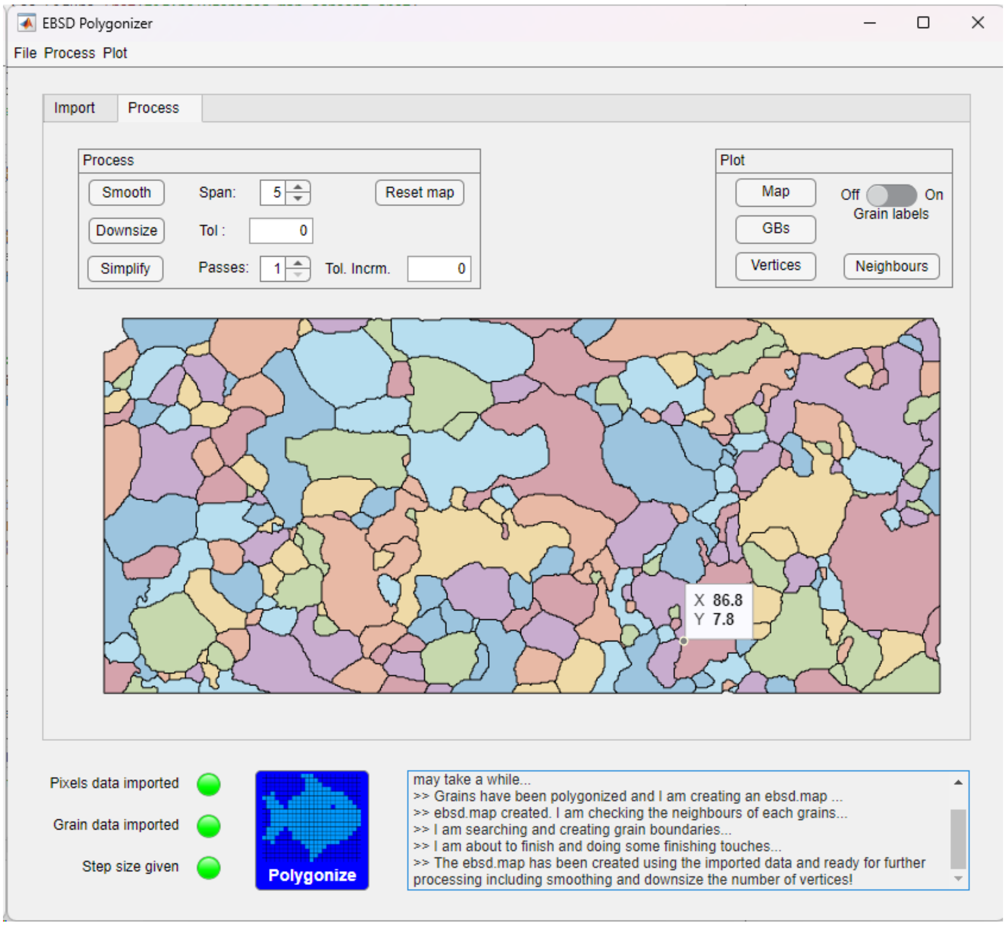


Figure 1.7: Polygonized microstructure

1.3.4 Post-processing

At this point, the EBSD data have been polygonized. An `ebsd.map` has been created. To facilitate FE modelling, post-processing functions like smoothing and downsizing are available to refine the polygonized EBSD data.

Smooth

As shown in Figure 1.8 the grain boundaries have a stair-case-like shape. Utilize the smoothing function to enhance the grain boundary representations, adjusting `span` parameters as necessary to achieve the desired level of detail. The figure shows the result for the default `span = 5`.

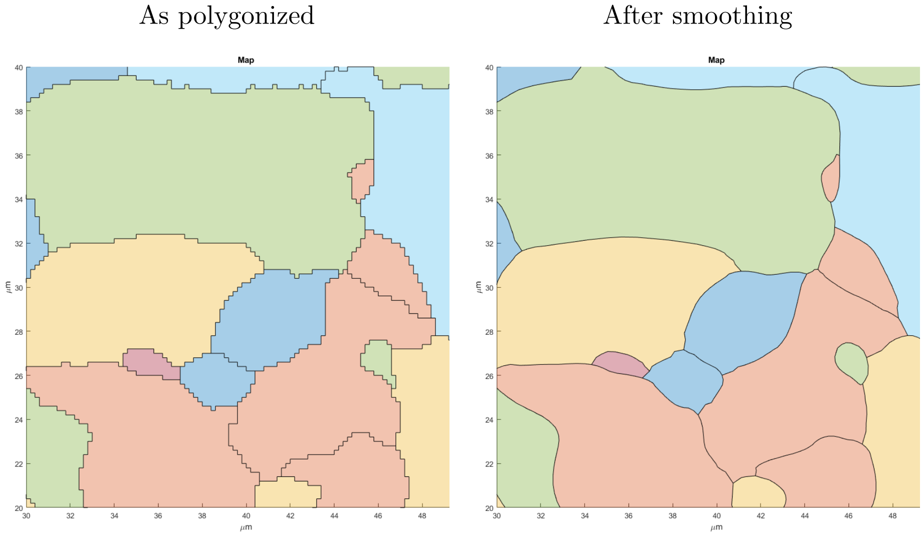


Figure 1.8: Smoothing the grain boundaries.

Downsize

The downsizing function allows for the reduction of vertices within the polygonized data, enabling more efficient FE modeling. Adjust tolerance levels to manage the level of detail retained in the downsize process. If the area of triangle made of three neighbouring vertices in a grain boundary segment is smaller than the tolerance, the middle vertice will be removed. Click on `Reset map` button to restore the original map. Figure 1.9 plot the map and vertices for different values of `downsize` tolerance.

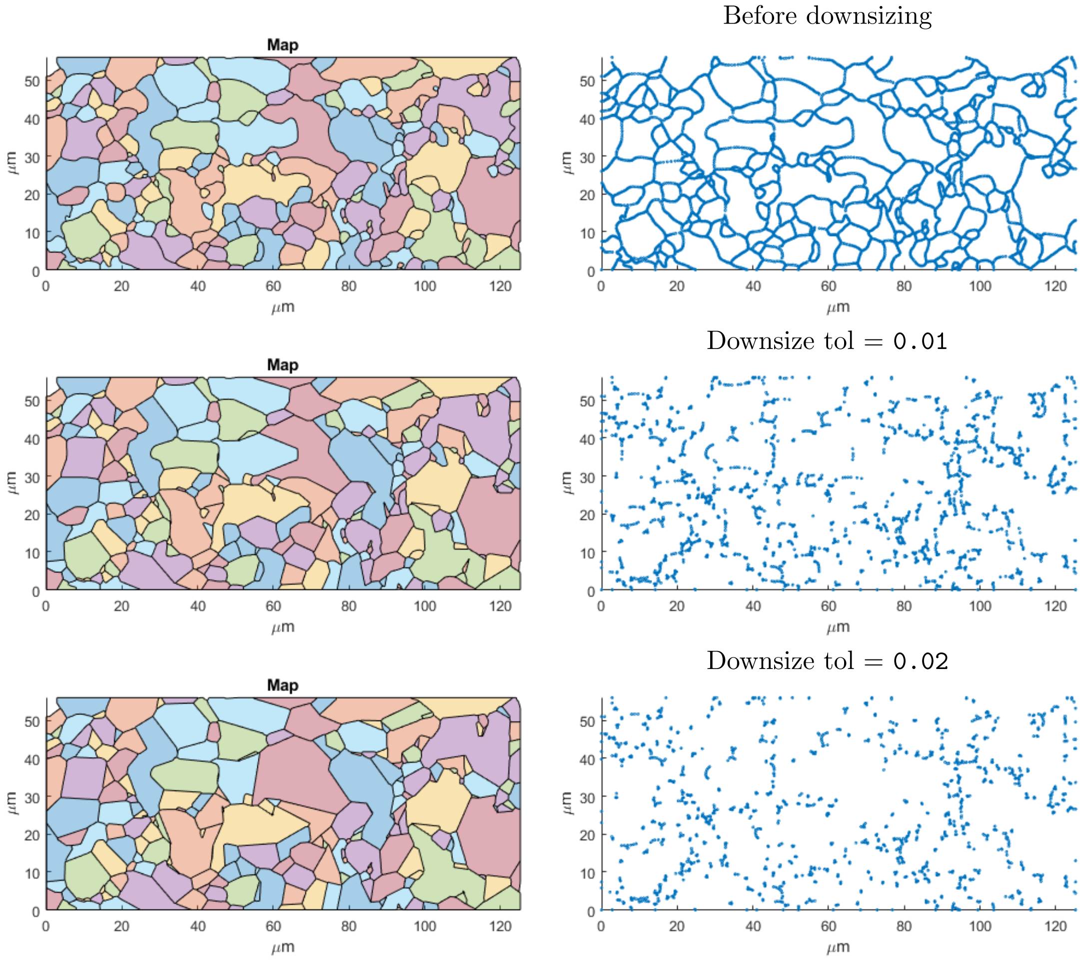


Figure 1.9: Downsize processing results.

Simplify

A combination of downsizing and smoothing, applied iteratively, can be used for more nuanced simplification of the dataset. Adjust the parameters including, `downsize tol`, number of passes, and the increment of tolerance each pass, `Tol. incrm.`. Figure 1.10 shows some typical result. The resultant number of vertices are shown in the log window.

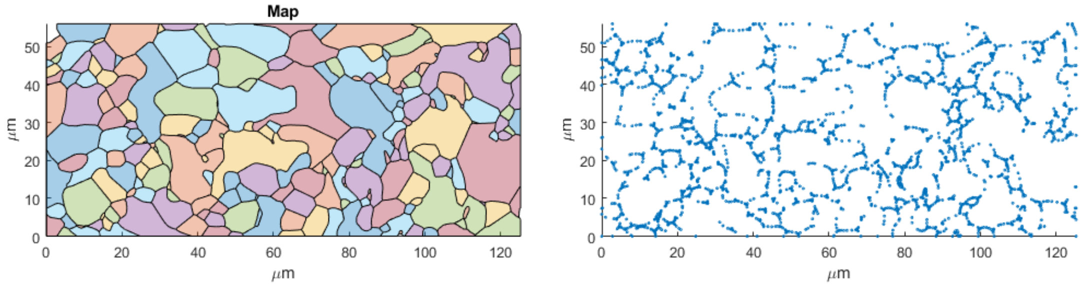


Figure 1.10: Simplified by 5 passes of `smooth` and `downsize`. `Downsize tol = 0.002, Tol. Incrm.=0.002`

1.3.5 Plot

The `EBSDPolygonizer` app encompasses a diverse range of plotting functionalities to visualize various aspects of the EBSD data effectively. These include the capability to plot the `ebsd.map`, neighbouring grains of a selected grain, grain boundaries, and vertices. Users have the option to augment the plot by including the Grain IDs.

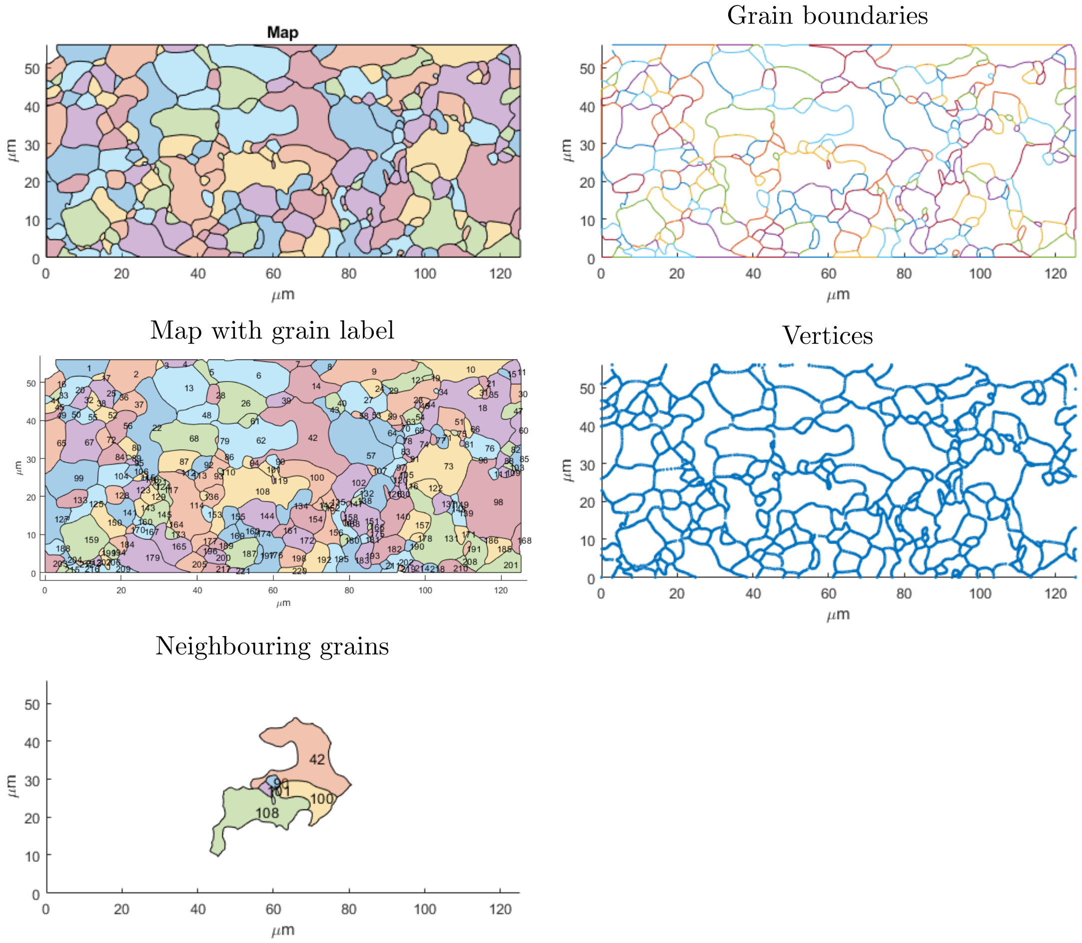


Figure 1.11: Plot functions

1.3.6 Save and Load

Save as file

Go to `File>Save>As MAT file` menu to save the post-processed data as an `ebsd.map` object within a MAT-file for future use. Enter the name of the variable or use the default `emap` and select where to save the MAT file.

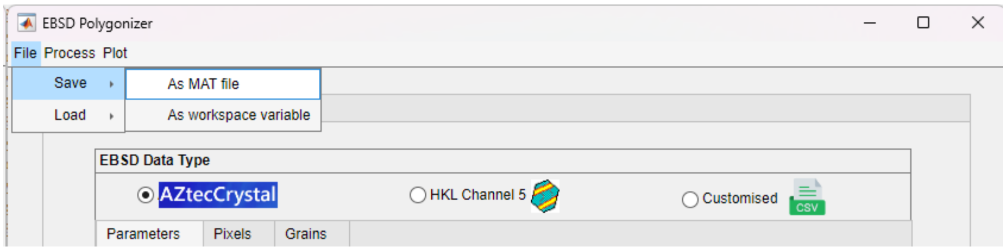


Figure 1.12: Save function in File Menu.

Save in MATLAB workspace

Go to `File>Save>As workspace variable` menu to export the post-processed map to the MATLAB workspace as an variable.

Load from file

Use the application's load function to reopen existing map for processing. Go to `File>Load>From MAT file` menu to load an `ebsd.map` from a MAT file.

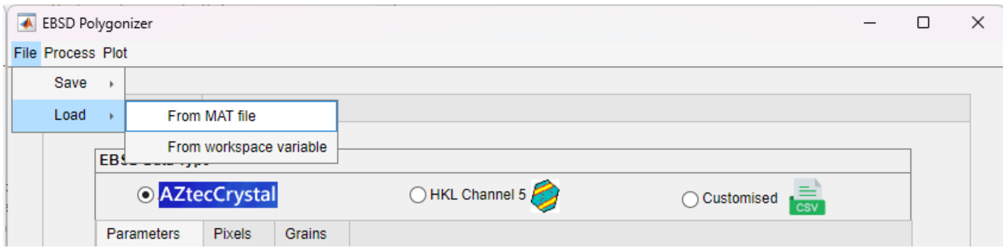


Figure 1.13: Load function in File menu.

Load from MATLAB workspace

Go to `File>Load>From workspace variable` menu to load an `ebsd.map` from base workspace.

1.3.7 `ebsd.map` in MATLAB

In MATLAB, load the saved `ebsd.map` object into the workspace. To view the properties of ebsd.map, type the name of the variable into the MATLAB command window as shown in Figure 1.14.

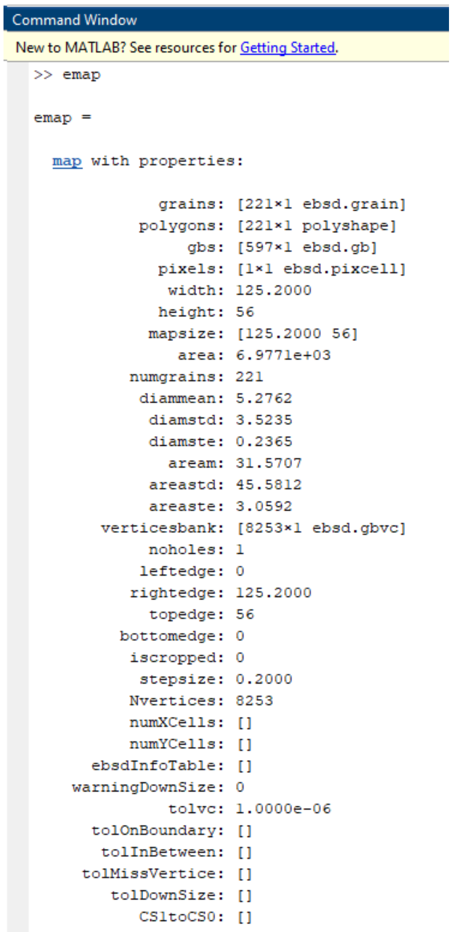
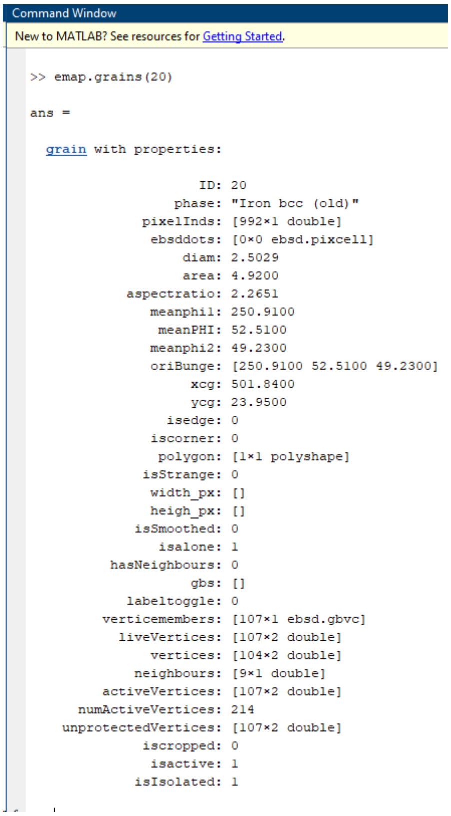


Figure 1.14: `ebsd.map` properties.

`ebsd.map` encompasses the microstructure information and the other data including grains (`ebsd.grain`), pixels (`ebsd.pixcell`), grain boundaries (`ebsd.gb`), etc.

The `ebsd.grain` object comprises data related to the corresponding polygons represented as `polyshape`. Within each ebsd.grain, you will find the coordinates of the vertices, grain orientation presented in Euler angles using Bunge notation, and IDs of neighboring grains. Figure 1.15 shows the properties of Grain 20.



```
Command Window
New to MATLAB? See resources for Getting Started.

>> emap.grains(20)

ans =

  grain with properties:

                     ID: 20
                  phase: "Iron bcc (old)"
              pixelInds: [992×1 double]
               ebsddots: [0×0 ebsd.pixcell]
                   diam: 2.5029
                   area: 4.9200
            aspectratio: 2.2651
               meanphi1: 250.9100
                meanPHI: 52.5100
               meanphi2: 49.2300
               oriBunge: [250.9100 52.5100 49.2300]
                    xcg: 501.8400
                    ycg: 23.9500
                 isedge: 0
               iscorner: 0
                polygon: [1×1 polyshape]
              isStrange: 0
               width_px: []
               heigh_px: []
             isSmoothed: 0
                isalone: 1
          hasNeighbours: 0
                    gbs: []
            labeltoggle: 0
         verticemembers: [107×1 ebsd.gbvc]
           liveVertices: [107×2 double]
               vertices: [104×2 double]
             neighbours: [9×1 double]
         activeVertices: [107×2 double]
      numActiveVertices: 214
    unprotectedVertices: [107×2 double]
              iscropped: 0
               isactive: 1
             isIsolated: 1
```

Figure 1.15: `ebsd.grain` properties.

The grain boundaries for the microstructure are stored under the `gbs` property as `ebsd.gb` type. For each grain boundary, the `ebsd.gb` object contains specific details such as the ID of the grains sharing the boundary and the grain boundary misorientation (`misori`). Figure 1.16 shows the properties of `ebsd.gb` for Grain boundary.

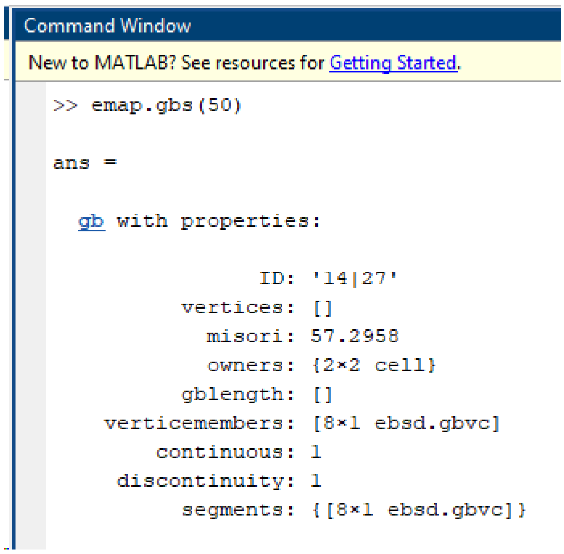


Figure 1.16: `ebsd.gb` properties