User Manual

IPSAT: Image based Particle Shape Analysis Toolbox

Contact information: Dr. Mohit Tunwal (mohit.tunwal@ucc.ie)

Dr. Kieran Mulchrone (k.mulchrone@ucc.ie)

Table of Contents

1. Set up

1.1 Mathematica installation

IPSAT requires Mathematica platform to run. Mathematica can be installed on Linux, Windows or Mac Operating System. The instructions for installation of Mathematica can be found in the following link: https://reference.wolfram.com/language/tutorial/InstallingMathematica.html IPSAT was developed on Mathematica version 10 and is compatible on version 11 as well. Once

1.2 Setting Directory

The IPSAT package ("IPSAT package.m") and example analysis notebook for both loose sediments ("Example Analysis Notebook - Loose sediment.nb") and thin section images ("Example Analysis Notebook – Thin Section.nb") can be downloaded from IPSAT folder. The folder containing IPSAT package and images to be processed must be stored in a single folder which is to be set as directory. To run the IPSAT package, a Mathematica notebook file is to be opened. This can be done in a new Mathematica notebook or in the example notebook files provided in the IPSAT folder. Running the following command with the respective directory location will set the directory:

$SetDirectory["C:\\Users\\user\\Desktop\\IPSAT\ folder"]$

1.3 Loading IPSAT package

The following command loads the IPSAT package ("IPSAT package.m") from the folder containing IPSAT package:

$<<\ "C:\\Users\\user\\Desktop\\IPSAT\ folder\\IPSAT\ package.m"$

```
(*Specify Directory where image files are located *)

In[1]:= SetDirectory["C:\\Users\\user\\Desktop\\IPSAT folder"]

Out[1]= C:\Users\user\Desktop\IPSAT folder

(*Load IPSAT package*)

In[2]:= << "C:\\Users\\user\\Desktop\\IPSAT folder\\IPSAT package.m"
```

Figure 1: Setting directory and loading IPSAT package

2. Image processing

2.1 Image input

The following command is used to load an input file:

$im = Import["filename"]$

The filename is to be replaced with the respective input image filename along with extension.

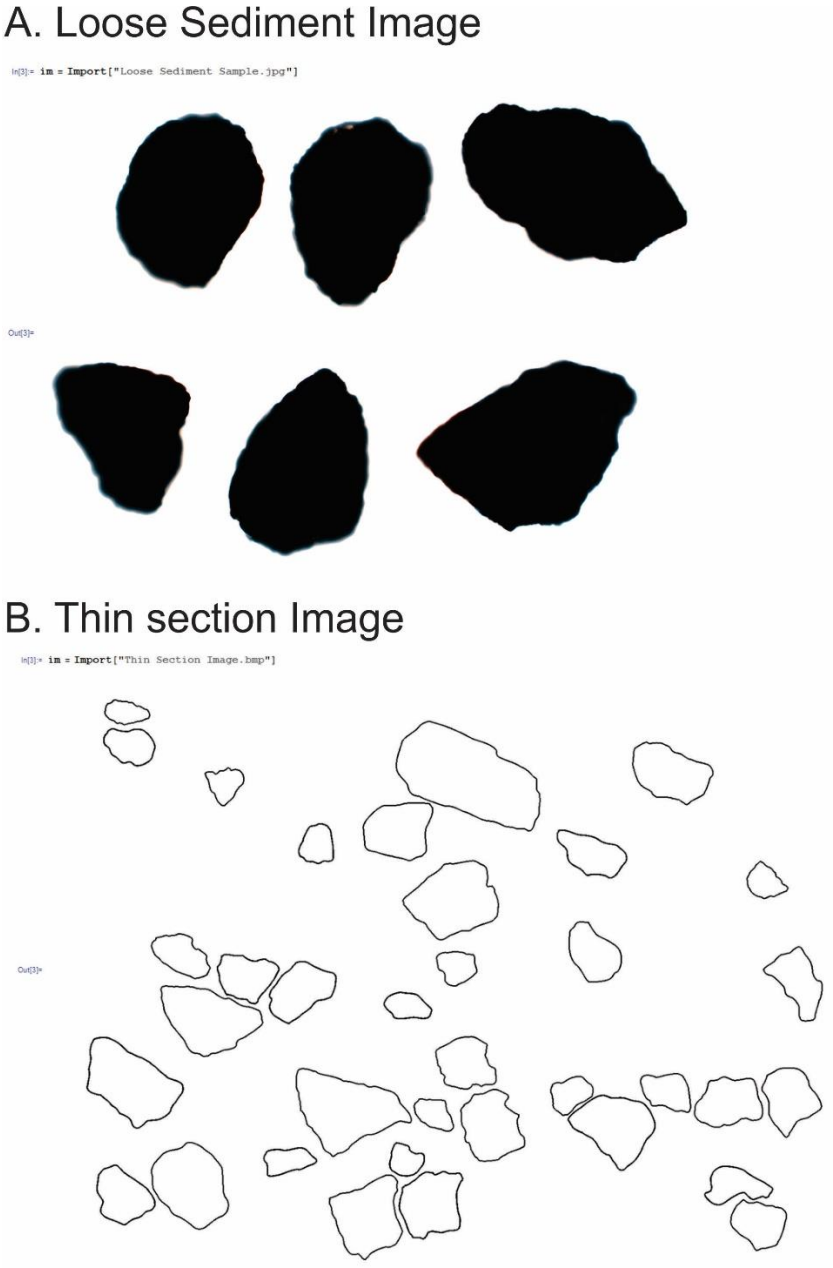


Figure 2: Loading an image for a) loose sediment microphotograph and b) manually traced thin section image

2.2 Boundary detection (applicable only for loose sediment microphotograph image)

The command GrainBoundary is used to apply threshold to loose sediment microphotograph image. The values 0.65 and 0.2 in the example image can changed to adjust threshold limit. In case of thin section sample, skip this step and proceed directly to next step. The command used for this step is:

$im = GrainBoundary[im, 0.65, 0.2]$

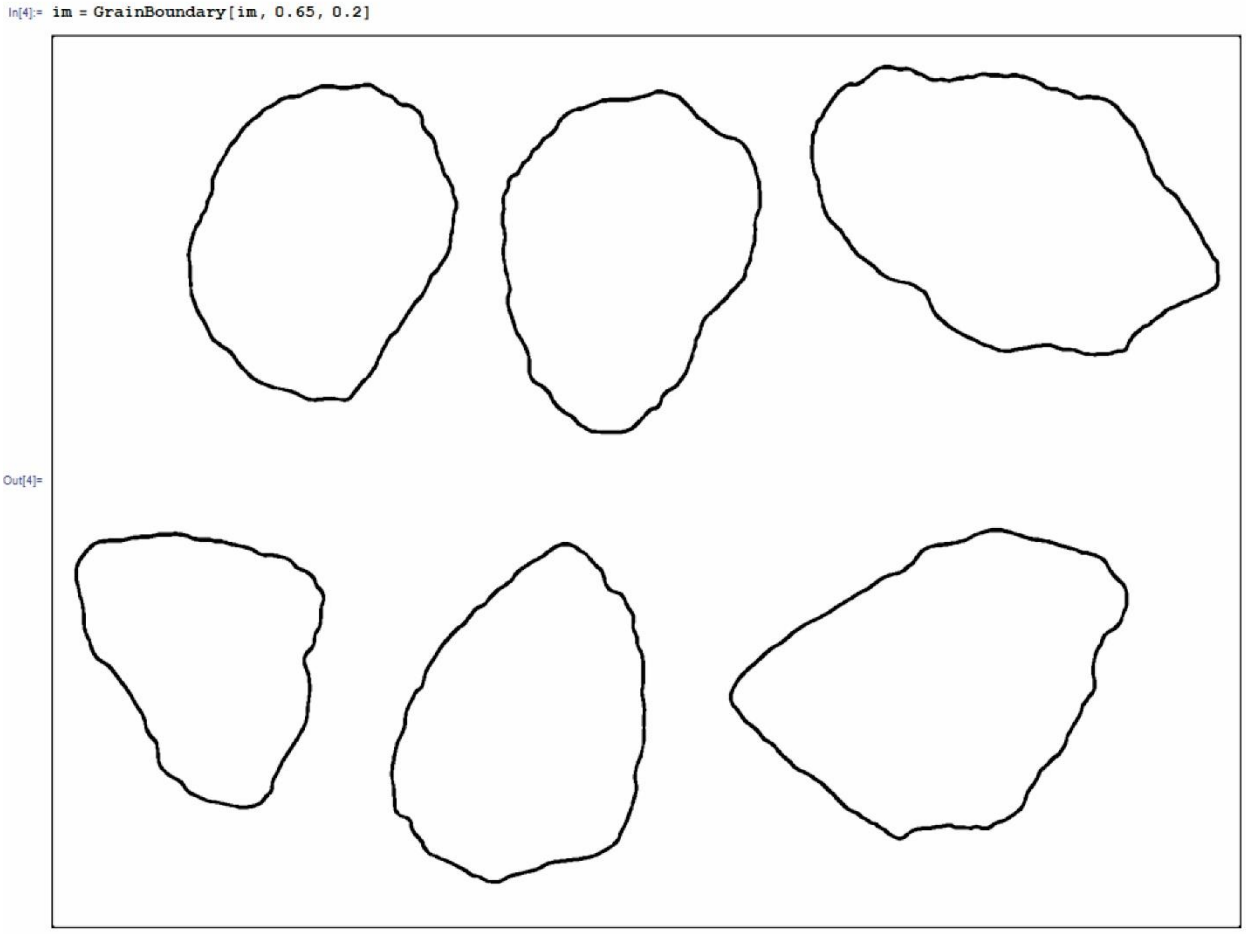


Figure 3: Use of GrainBoundary function to detect particle boundary

Note that the output of this step is similar to input of manually traced input image for thin section sample.

2.3 Particle region identification

GrabImage is for identifying individual particle region(s) in the image. From this step, all the following steps are applicable to both loose sediment and thin section analysis. The command is as following:

$res1 = GrabImage[im];$

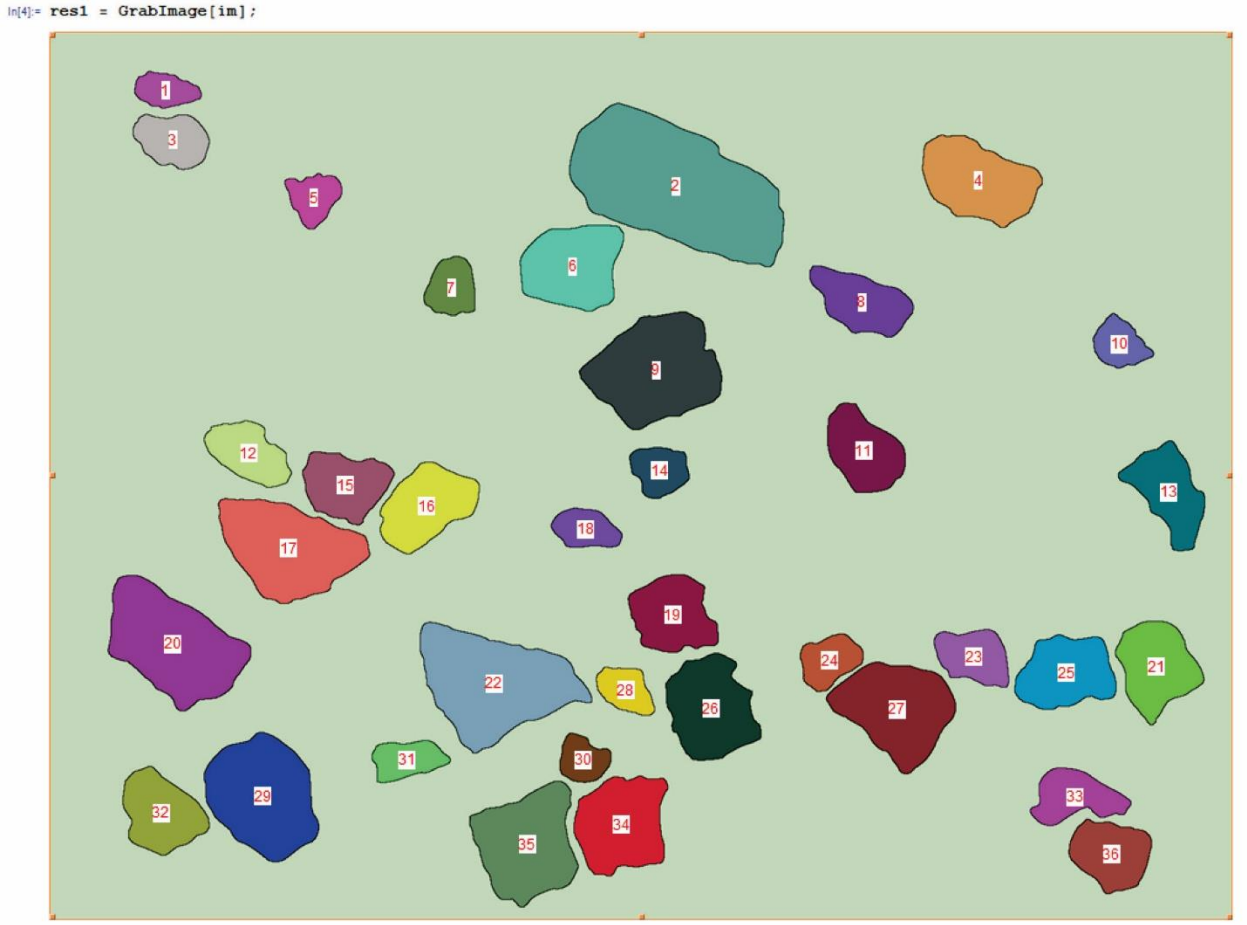


Figure 4: GrabImage function to identify particle regions

Note that this step may take some run time. The labels on the particle show label number of the corresponding identified region as a particle. The colour is randomly allotted to the particles.

2.4 Removal of unwanted regions

The function RefineImage is to remove any unwanted region in the image resulting from previous step. This could be due to either unwanted artefact present in the image or a bounded region erroneously identified as a particle. To remove such regions, the corresponding labels can be fed into RefineImage function as:

$res2 = RefineImage[res1, \{1,12,4\}];$

In this example, particle with label number 1, 12 and 4 are fed in the RefineImage function. On running the command, the output image will remove the given labels from the previous image.

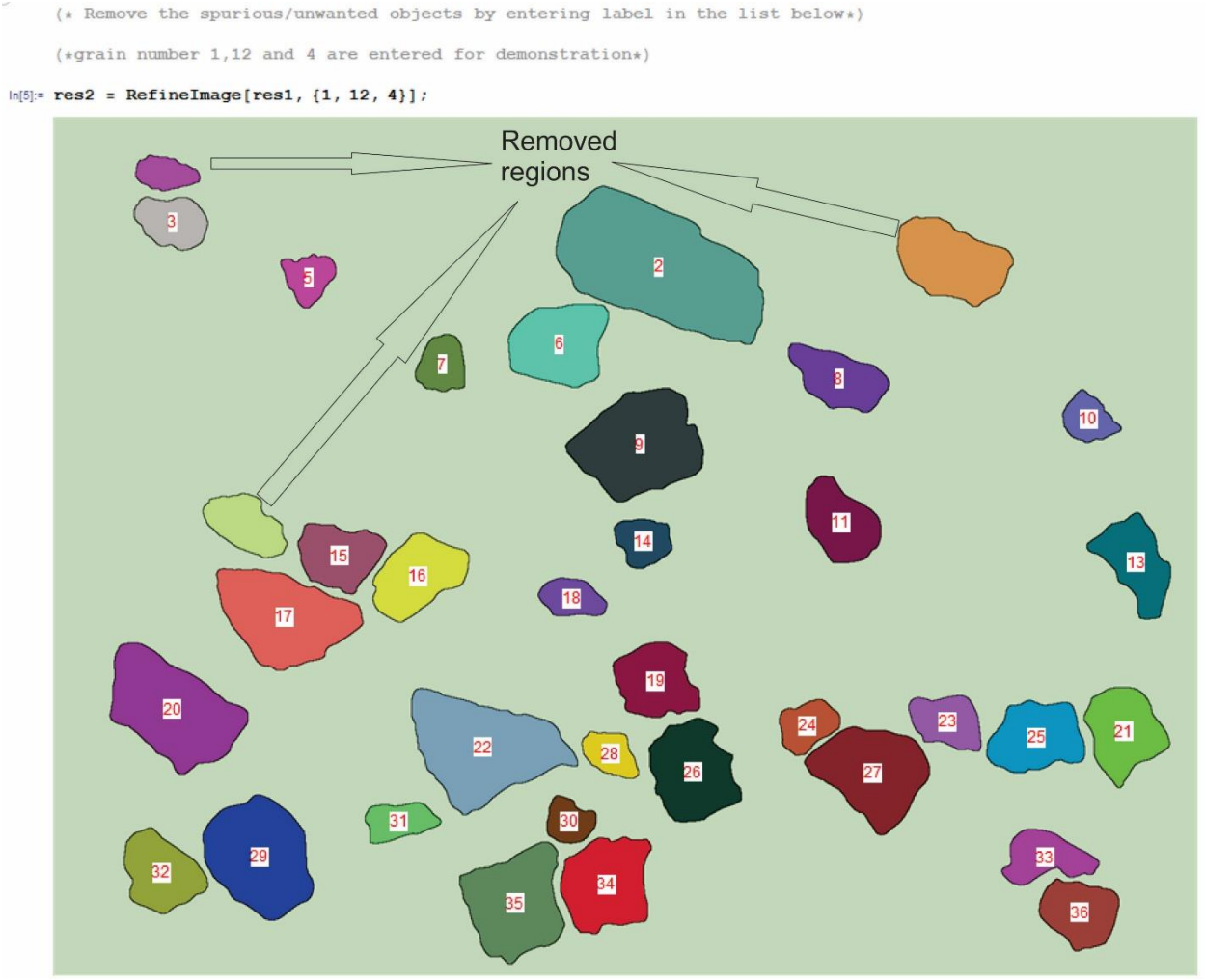


Figure 5: RefineImage function removes particle region with 1,12 and 4 label

Note that if there is no spurious region to remove then the command should be run with the label number field left empty as:

$res2 = RefineImage[res1, \{\}];$

3. Data extraction

The function ExtractData consolidates all the data extracted from the previous step so that it can be used for shape calculation in coming steps. Two more functions CircumsribedCircle and InscribedCircle calculates smallest circumscribing circle and largest inscribing circle for the particle boundary. The commands are as follows:

$data = ExtractData[res2];$

$listcircum = CircumscribedCircle[data];$

$listinrad = InscribedCircle[data];$

```
    (* Extract boundary coordinates,
    inside coordinates and some geometric measurements of grains *)
In[6]:= data = ExtractData[res2];
    (* Calculates the largest inscribed circle and smallest inscribed circle for
     each grain *)
In[7]:= listcircum = CircumscribedCircle[data];
In[8]:= listinrad = InscribedCircle[data];
```

Figure 6: ExtractData, CircumscribedCircle and InscribedCircle function to prepare data for further shape parameter calculation

4. Parameter Calculation

4.1 Shape parameter calculation

The following commands calculates fractal dimension, roundness, angularity, irregularity and circularity parameters. Based on choice of operator, a selection of parameters or all the parameters can be calculated. The functions are as follows:

$listfractal = FractalDivider[data];$

$listirregularity = Irregularity[res1, data];$

$listroundness = Roundness[data, listinrad];$

$listangularity = Angularity[data, 50{,}5];$

$listcircularity = CircularityFunction[listcircum, listinrad];$

For Angularity function, the second and third arguments are the number of vertices in polygon and the number of highest differences of angles used for angularity calculation respectively (refer to manuscript for details). The example notebook here uses 50 and 5 values which can be changed according to users' requirement.

```
(* Calculates the fractal divider for each grain *)

In[9]:= listfractal = FractalDivider[data];

(* Calculates the irregularity for each grain *)

In[10]:= listirregularity = Irregularity[res1, data];

(* Calculates the roundness for each grain *)

In[11]:= listroundness = Roundness[data, listinrad];

(* Calculates the angularity for each grain.*)
(* In second field enter the number of sides of polygon each grain has to be
 turned into.*) (* In this example all the grains are turned into 50 sided polygon.*)
(* In third field shows the number of highest differences of angles to be
 averaged for Angularity.*)

In[12]:= listangularity = Angularity[data, 50, 5];

(* Calculates the circularity for each grain *)

In[13]:= listcircularity = CircularityFunction[listcircum, listinrad];

(*List of fourier descriptors. The second arguement takes in the number of
 descriptors required for each grain*)

In[14]:= listfourier = FourierDescriptors[data, 120];
```

Figure 7: FractalDivider, Irregularity, Roundness, Angularity, Circularity and FourierDescriptor functions for parameter calculation

Fourier Descriptors are calculated using the following command:

$listfourier = FourierDescriptors[data, 120];$

The second argument (120 in this example) is the number of fourier descriptor per particle boundary required by the operator. Like other parameters, this function is also optional to use.

Other parameters from Table 1 of the manuscript are to be calculated in subsequent step during result formatting in table (see section 5.1)

4.2 Size calculation

The third argument of SizeData function requires width of the input image to translate pixel dimension to either microns or mm or any other length unit. In this example, the actual width of the input image is 2186 microns. Hence, the subsequent result in the size data will be in microns.

```
(* Calculates the size for each grain. Enter the actual width of image below
 in unit of your choice. In this example it is in microns *)

In[15]:= listsizedata = SizeData[data, listcircum, 2186];
```

Figure 8: Size calculation using SizeData Function. 2186 is the width of input image in microns

5. Results

5.1 Result Table

The function ResultTable organises the results in a tabular format for export. The second argument takes in the list of shape parameters to be displayed (and exported in the next step). The users have a choice to select the parameters to be fed into this list. The third argument is for the Other Parameters (refer to Table 1 in the manuscript). It takes "True" or "False" as an input. The fourth argument is size data. It can be kept empty to leave out size results.

$$r1 = ResultTable[data, \{listroundness, listcircularity, listangularity, listirregularity, listfractal\}, True, \{listsizedata\}, listinrad]$$

```
(*Displays the result output. The second arguement takes in the list of shape
 parameters required to be displayed and exported*)
(*The third arguement takes input as either True or False to display Other
 shape parameters*)

(*The fourth argument takes in the size data. It can be left empty like {} if
 size is not required*)

In[16]:= r1 = ResultTable[data, {listroundness, listcircularity, listangularity,
    listirregularity, listfractal}, True, {listsizedata}, listinrad]

Out[16]//TableForm=
```

Label	Roundness	Circularity	Angularity	Irregularity	Fractal Divider	Aspect Ra
2	0.640842	0.686313	48.6397	0.118449	1.01439	2.12325
3	0.596773	0.788251	46.9786	0.127366	1.01692	1.51818
5	0.568938	0.782854	65.4959	0.257477	1.04064	1.08826
6	0.531159	0.811769	42.6541	0.159691	1.01898	1.23627
7	0.589111	0.845839	44.7331	0.145601	1.01856	1.20054
8	0.592831	0.670688	48.2185	0.178104	1.01769	2.01973
9	0.550575	0.812539	65.6262	0.141973	1.04252	1.27042
10	0.597884	0.824619	73.1467	0.167608	1.04445	1.28221
11	0.656518	0.792605	43.7102	0.153573	1.01365	1.53545
13	0.558566	0.69017	53.1862	0.272208	1.03969	1.87383
14	0.608508	0.835108	51.7612	0.15948	1.02989	1.16801
15	0.590128	0.788808	55.3107	0.187158	1.02612	1.31976
16	0.553115	0.74129	46.6357	0.158157	1.02617	1.56863
17	0.576948	0.723673	60.7022	0.21504	1.02079	1.60457
18	0.591798	0.721975	33.2175	0.129058	1.01462	1.84531
19	0.529404	0.824816	74.7371	0.164752	1.04822	1.16625
20	0.572709	0.730871	51.7848	0.181177	1.04364	1.72185

Figure 9: The output from the ResultTable displaying all the parameters. Parameters right of Aspect Ratio are cropped out in the image

The above command displays all the results. The command below will display only Roundness, Circularity, Angularity and Irregularity results. It will omit Fractal Dimensions, Other Parameters and Size parameters:

$$r1 = ResultTable[data, \{listroundness, listcircularity, listangularity, listirregularity\}, False, \{\}, listinrad]$$

5.2 Export Result Table

The result displayed in the previous step can be exported to either excel or a csv file by the following command:

$$Export["filename", r1]$$

The filename includes the filename along with the preferred file format extension. The file is exported directly to the directory set in section 1.2.

```
(* Export the result in Excel format. Give the name of the file name to be
 generated. It will be saved in the directory selected *)

In[17]:= Export["Example Thin Section Res.xls", r1]

Out[17]= Example Thin Section Res.xls
```

Figure 10: Export function saving the file "Example Thin Section Res.xls" in excel format.

5.3 Display Grain Map

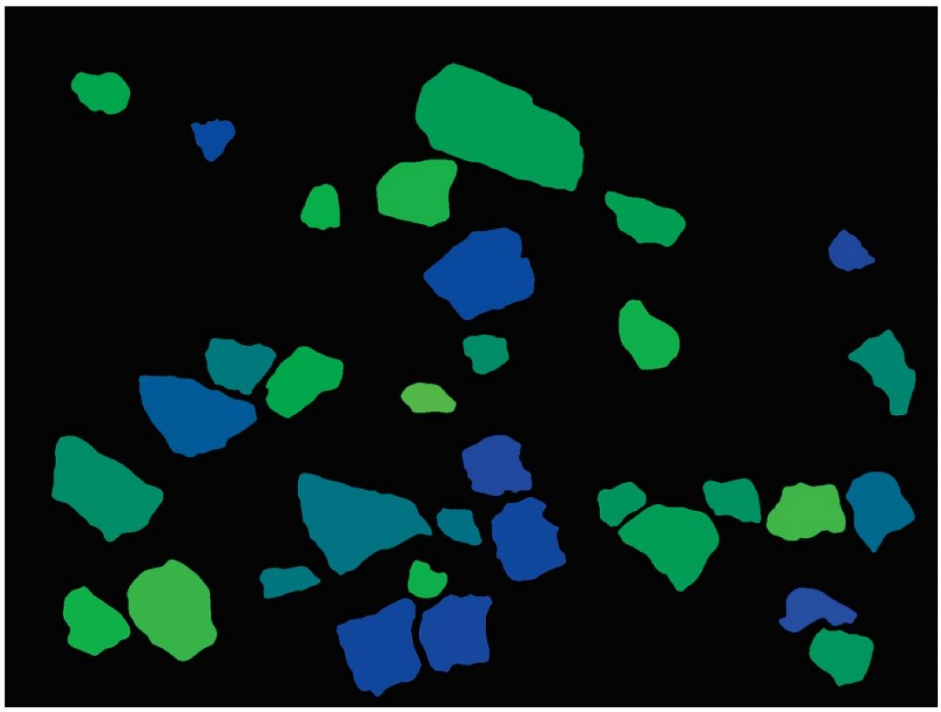
The function GrainMap displays particles with varying colour intensity based on the parameter value. The parameter (for example, *listroundness* for Roundness, *listangularity* for Angularity and likewise) is fed as the second argument. The third argument takes in the lower and upper threshold (30 and 80 in this example). All the particle having parameter value below lower threshold will be displayed by lower colour scheme and likewise for the upper threshold. The last argument takes in the colour for lower and upper threshold. The example in the command below and in Figure 11 has "Green" and "Blue" but any alternative colour scheme can be selected by the end user.

$$GrainMapping[res2, listangularity, \{30,80\}, \{Green, Blue\}]$$

Another example of command to generate a grain map for variation in roundness with 0.4 to 0.9 threshold values along with Yellow and Red lower and upper colour scheme would be:

$$GrainMapping[res2, listangularity, \{0.4,0.9\}, \{Yellow, Red\}]$$

```
In[18]:= GrainMapping[res2, listangularity, {30, 80}, {Green, Blue}]
```



Out[18]=

Figure 11: Grain map from bright green to dark blue for 30 and 80 lower and upper threshold values respectively

5.4 Export Fourier Descriptor

The ExportFourier function exports the Fourier Descriptor result to the location of directory. The third argument takes the filename for the exported file. Similar to the section 5.2, the results can be exported in excel or csv file format based on user preference.

$FourierOutput[listfourier, data, "filename"]$

```
(*output fourier descriptors*)

In[19]:= FourierOutput[listfourier, data, "fourier result.xls"]

Out[19]= fourier result.xls
```

Figure 12: FourierOutput function exporting the result as "fourier result.xls" file to the set directory

5.5 2D to 3D transformation

The function SizeTransform uses the STRIPSTAR algorithm described in Heilbronner and Barrett, 2014 to transform 2D size data to 3D size data for thin section images. The first argument of the SizeTransform function requires the user to input bin size for histogram calculation. The example bin size chosen in this case is 60 microns. The choice of 2D size parameter from Table 2 of the manuscript can be entered as a numeral code (1 for S_c, 2 for S_p, 3 for S_d, 4 for S_a, 5 for S_b and 6 for S_m) in the third argument.

$$threeDres = SizeTransform[60, listsizedata, 2]$$

$$Export["filename", threeDres]$$

The export function is then used to export the result of SizeTransform in a tabular format to the directory.

```
(*The first arguement takes in the class width for each bin. Third arguement
 takes in the numeral code for the size parameter from Table 2*)

(*r is the upper limit of each class, h(r) is the 2D size distribution*)

(*sph represents results for only positive spheres data,
whereas sph&asphs represents results for spheres as well as antispheres*)

(*h(R) represents number weighted distribution of 3D grains whereas v(R)
 represents volume weighted distribution*)

(*see chapter 12 of Heilbronner and Barrett, 2014 for details*)

In[20]:= threeDres = SizeTransform[60, listsizedata, 2]

Out[20]//TableForm=
r     h(r)       sph h(R)    sph v(R)    sph&asph h(R)    sph&asph v(R)
60    0.         0.          0.          -9.79305         -0.132763
120   3.0303     0.          0.          -13.6581         -1.48129
180   30.303     27.3175     8.01951     20.6752          7.56789
240   39.3939    47.6385     33.1498     36.0551          31.2829
300   12.1212    9.42368     12.8078     7.1323           12.0865
360   12.1212    12.7188     29.8705     9.62622          28.1884
420   0.         0.          0.          -0.863968        -4.01747
480   3.0303     2.90152     16.1524     2.19601          15.2428

(*Export 3d distribution results*)

In[21]:= Export["threeDres.xls", threeDres]

Out[21]= threeDres.xls
```

Figure 13: SizeTransform function transform 2D size data (S_p) to 3D size data. The bin size chosen is 60 microns. The result is exported to "threeDres.xls" file in the directory