
Table of Contents

.....	1
Load Toolboxes	1
h5oina file location	1
make a results folder if needed	2
load the h5oina file	2
Set the plotting preferences - this has to be validated for your instrument	2
Now do some plots	2
Extract the phase to plot	3
Plot the map of phases/indexed data	3
IPF-x, y and z	6
Plot the PF - scatter	10
Plot the IPF - scatter	11
Construct the ODF	11
Plot the uncorrelated misorientation distribution	13
Calculate grains and grain boundaries	14
Plot the grain size distributions	17
Filter the grain size data	20
Demonstrate that we can extract a point from this map	22
Extract a subset of the map	26
For this smaller region, plot all the IPFs	28
extract the grain data from the big map that fits within this smaller map	30
Plot a subset map with the unit cells over the top	32
Select an individual grain for texture based segmentation	34
plot some unit cells for this	38
copy this m file over, for archival purposes	40

```
clear;
home;
close all;
```

Load Toolboxes

The code requires MTEX version 5.11.2: <https://mtex-toolbox.github.io/download> (also works with 5.10.2)

```
%Toolboxes - folder locations
mtex_location='C:\Users\ruthb\OneDrive\Documents\MTEX\mtex-5.11.2'; % update
this

%start mtex if needed
try EBSD;
catch
    run(fullfile(mtex_location, "startup.m"));
end
```

h5oina file location

```
% folder
file1_folder='C:\Users\ruthb\DocumentsOnLaptop\GitHub\MTEX_Workshop
\Data'; %location where the data is stored
```

```
% file
file1_name='Mg_Pecs1 Specimen 1 Site 1 Map Data 1'; %should be a h5oina file,
do not add in the .h5oina file extension

%extra (workshop)
file_dset='1'; %data set number of interest in the h5 file
mb_length = 50; %micro bar length for plots - if you want to override, you can
comment this out/clear this variable and the override will not happen
```

make a results folder if needed

```
%create a results folder
file1_name_us=file1_name;
file1_name_us(strfind(file1_name_us, ' '))=' '_;

resultsdir=fullfile(cd,'results',file1_name_us);
if isdir(resultsdir) == 0; mkdir(resultsdir); end
```

load the h5oina file

```
% Make a full file name
file1_full=fullfile(file1_folder,[file1_name '.h5oina']);

% load one file
% h5oina_file=file1_full; % file name
warningOn=0; % turn on/off warnings during loading
[ebsd_original,dataset_header,ebsd_patternmatched,h5_original,h5_patternmatch]
= load_h5oina_pm2(file1_name,file1_folder,warningOn);
ebsd=ebsd_patternmatched; % if not pattern matched it will default to original
ebsd data
ebsd=ebsd.gridify;
```

Mg_Pecs1 Specimen 1 Site 1 Map Data 1 - ebsd data loaded

Set the plotting preferences - this has to be validated for your instrument

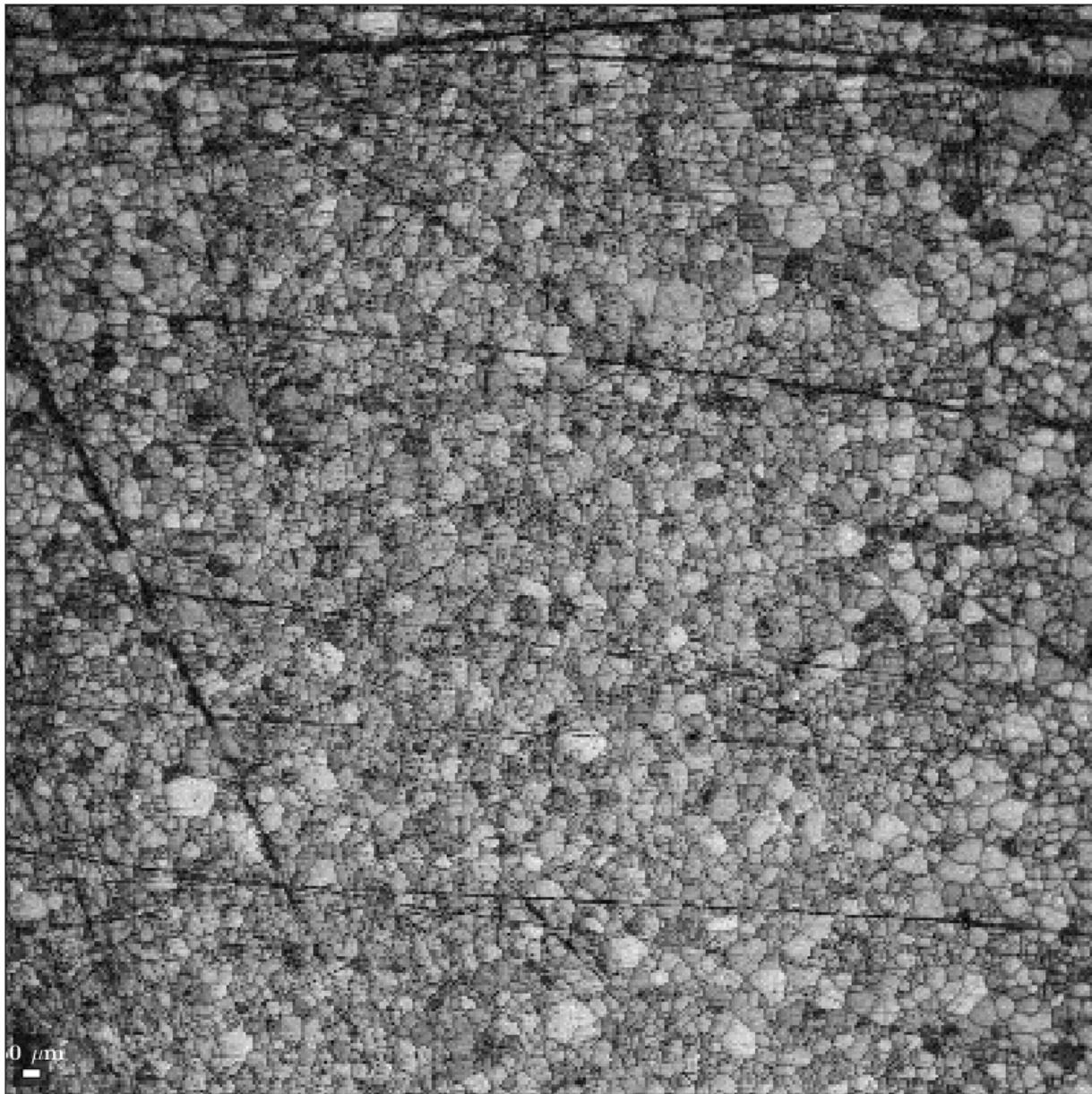
```
%this is set up for the pFIB and Oxford Insturments systems at UBC
setMTEXpref('xAxisDirection','east'); %aztec
setMTEXpref('zAxisDirection','outofPlane'); %aztec

%font size and overwrite the scale bar if you want
setMTEXpref('FontSize',12)
```

Now do some plots

```
%band contrast
figure;
plot(ebsd,ebsd.prop.Band_Contrast); colormap('gray');
```

```
if exist('mb_length','var'); mp = MTEX_mb_fix(mb_length); end %change the  
micronbar length to make it prettier/easier to read
```



Extract the phase to plot

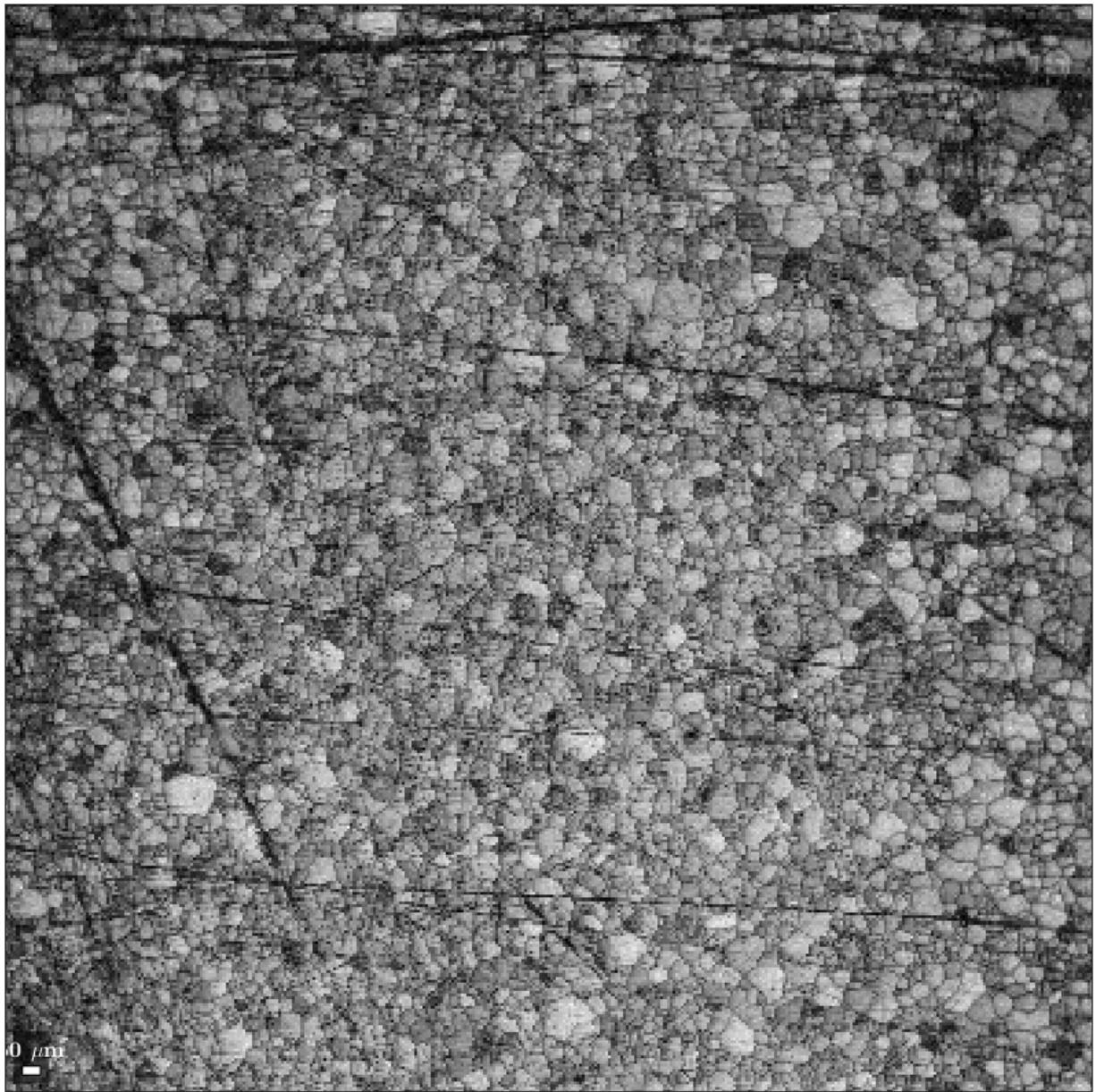
```
%select the mineral name  
mineral_name=ebsd.CS.mineral;  
% mineral_name='Magnesium';
```

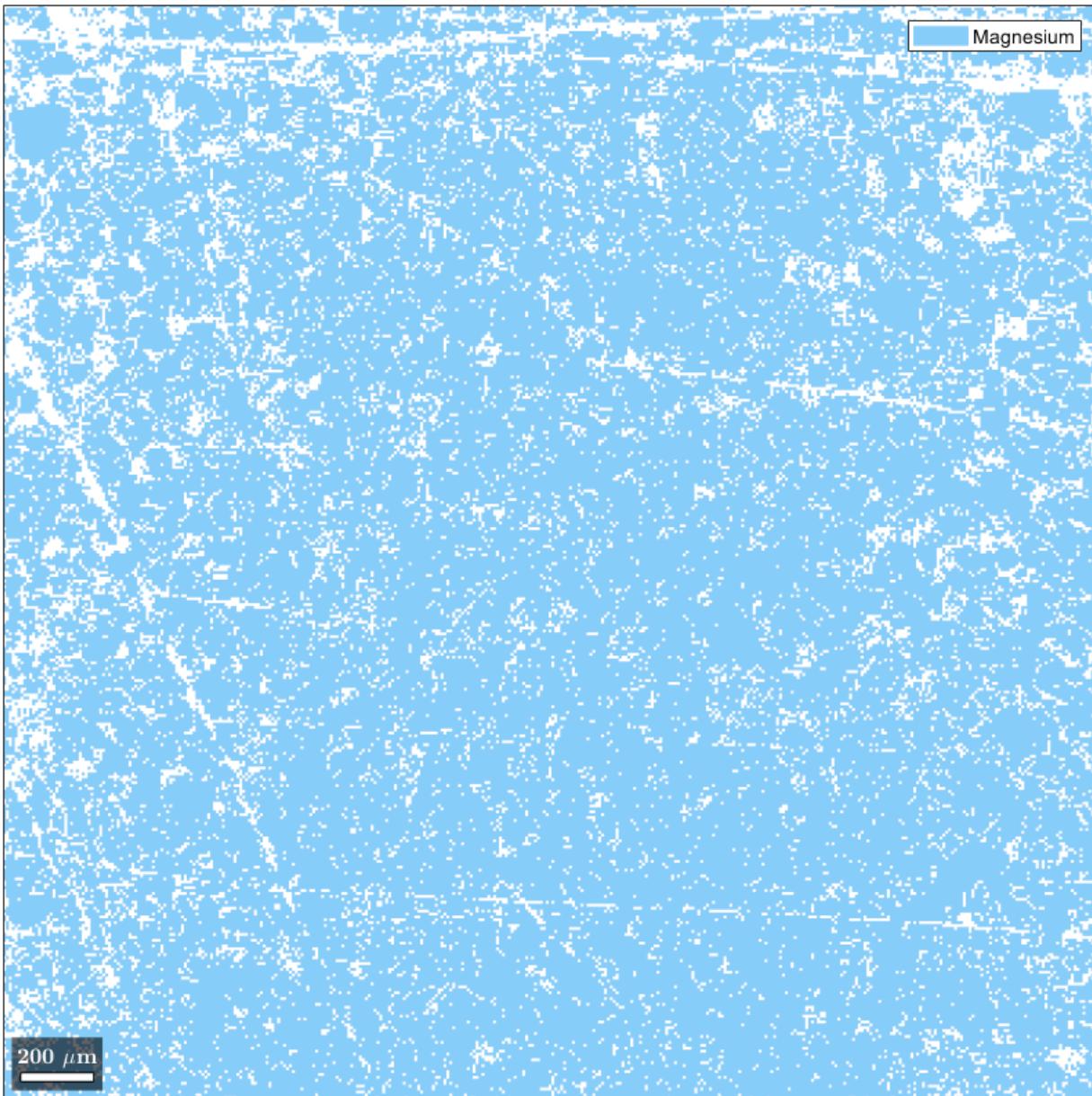
Plot the map of phases/indexed data

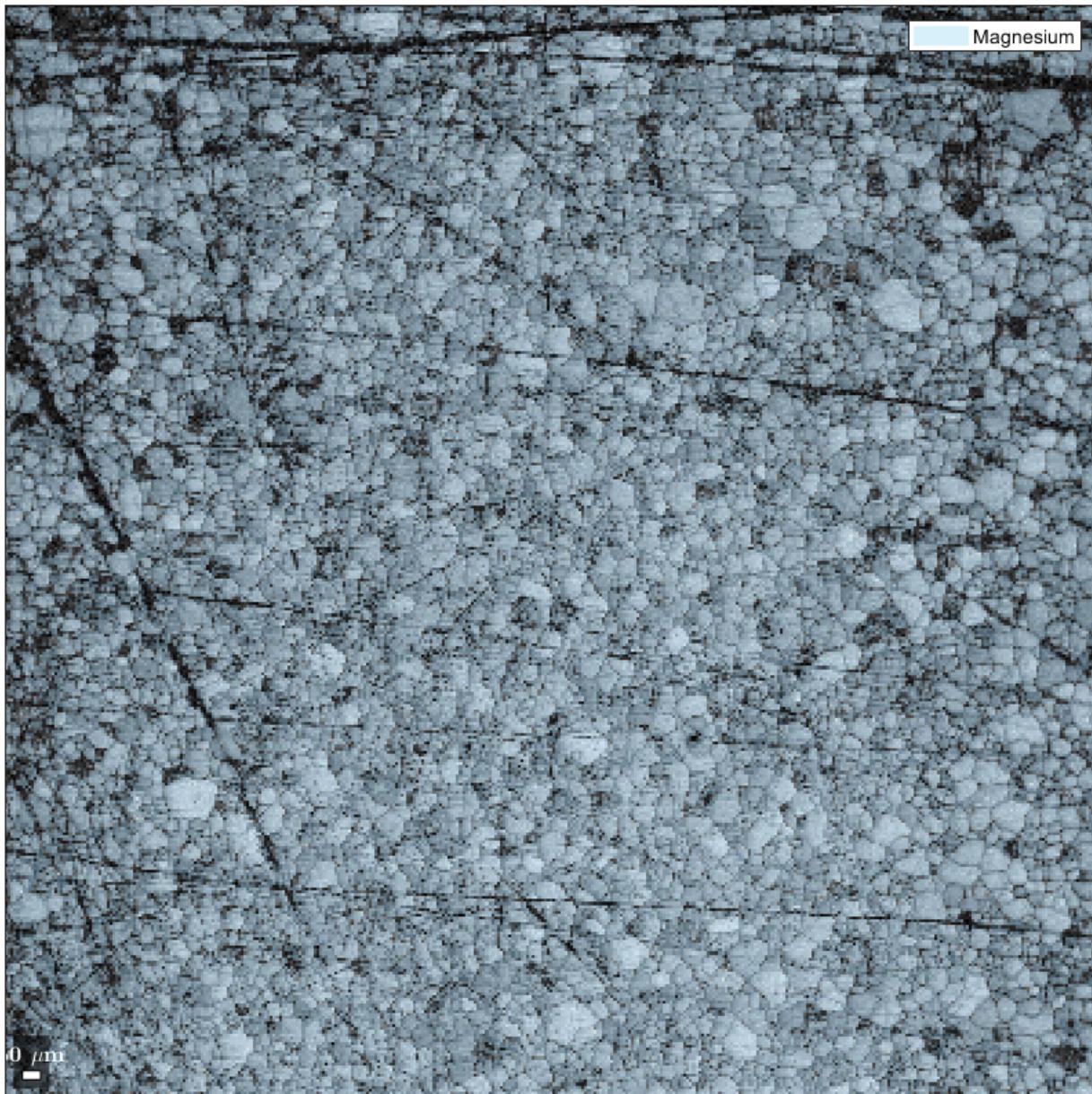
```
figure;
```

```
plot(ebsd');

%overlay with Band Contrast
figure;
plot(ebsd,ebsd.prop.Band_Contrast); colormap('gray');
hold on;
plot(ebsd,'FaceAlpha','0.3');
hold on;
if exist('mb_length','var'); mp = MTEX_mb_fix(mb_length); end %change the
micronbar length to make it prettier/easier to read
```







IPF-x, y and z

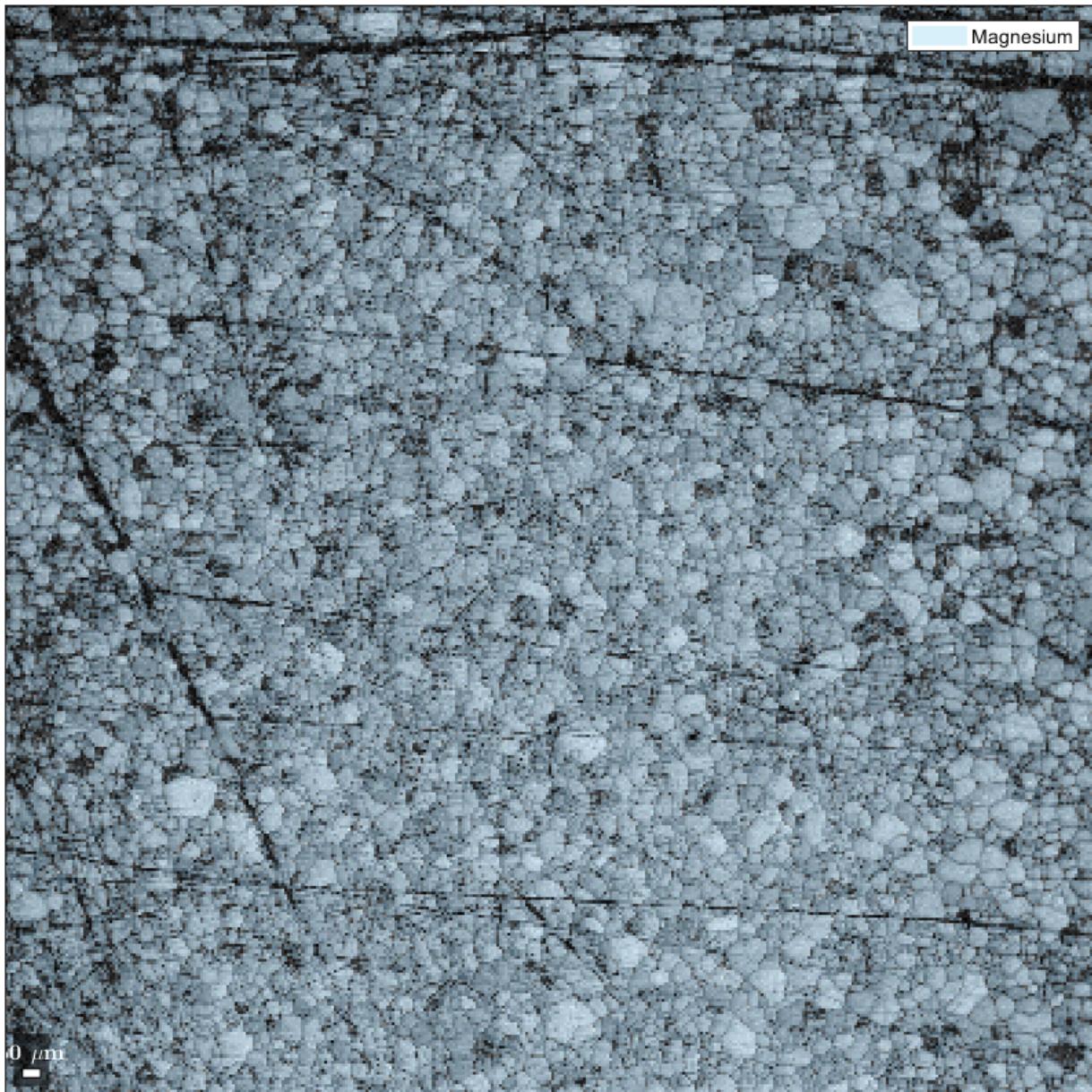
```
compute the colors ipfKey = ipfHSVKey(ebsd(mineral_name));  
  
ipfKey = ipfTSLKey(ebsd(mineral_name)); %use the TSL  
  
%plot the key  
f1=figure;  
plot(ipfKey)  
  
% now generate the IPF colour maps  
ipfKey.inversePoleFigureDirection = vector3d.X;  
colors_X = ipfKey.orientation2color(ebsd(mineral_name).orientations);
```

```
ipfKey.inversePoleFigureDirection = vector3d.Y;
colors_Y = ipfKey.orientation2color(ebsd(mineral_name).orientations);
ipfKey.inversePoleFigureDirection = vector3d.Z;
colors_Z = ipfKey.orientation2color(ebsd(mineral_name).orientations);

%now plot the three IPF coloured maps
f2=figure;
plot(ebsd(mineral_name),colors_X,'micronbar','off');
nextAxis
plot(ebsd(mineral_name),colors_Y,'micronbar','off');
nextAxis
plot(ebsd(mineral_name),colors_Z,'micronbar','on');

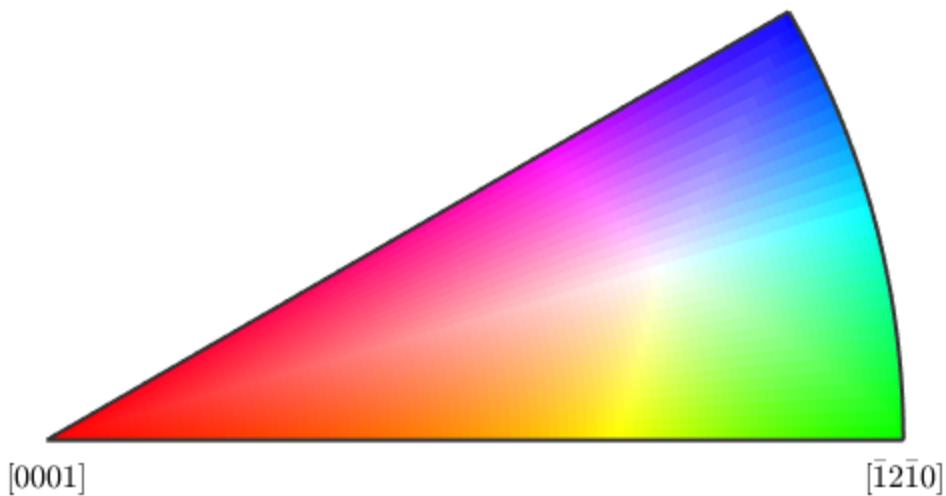
if exist('mb_length','var'); mp = MTEX_mb_fix(mb_length); end %change the
micronbar length to make it prettier/easier to read

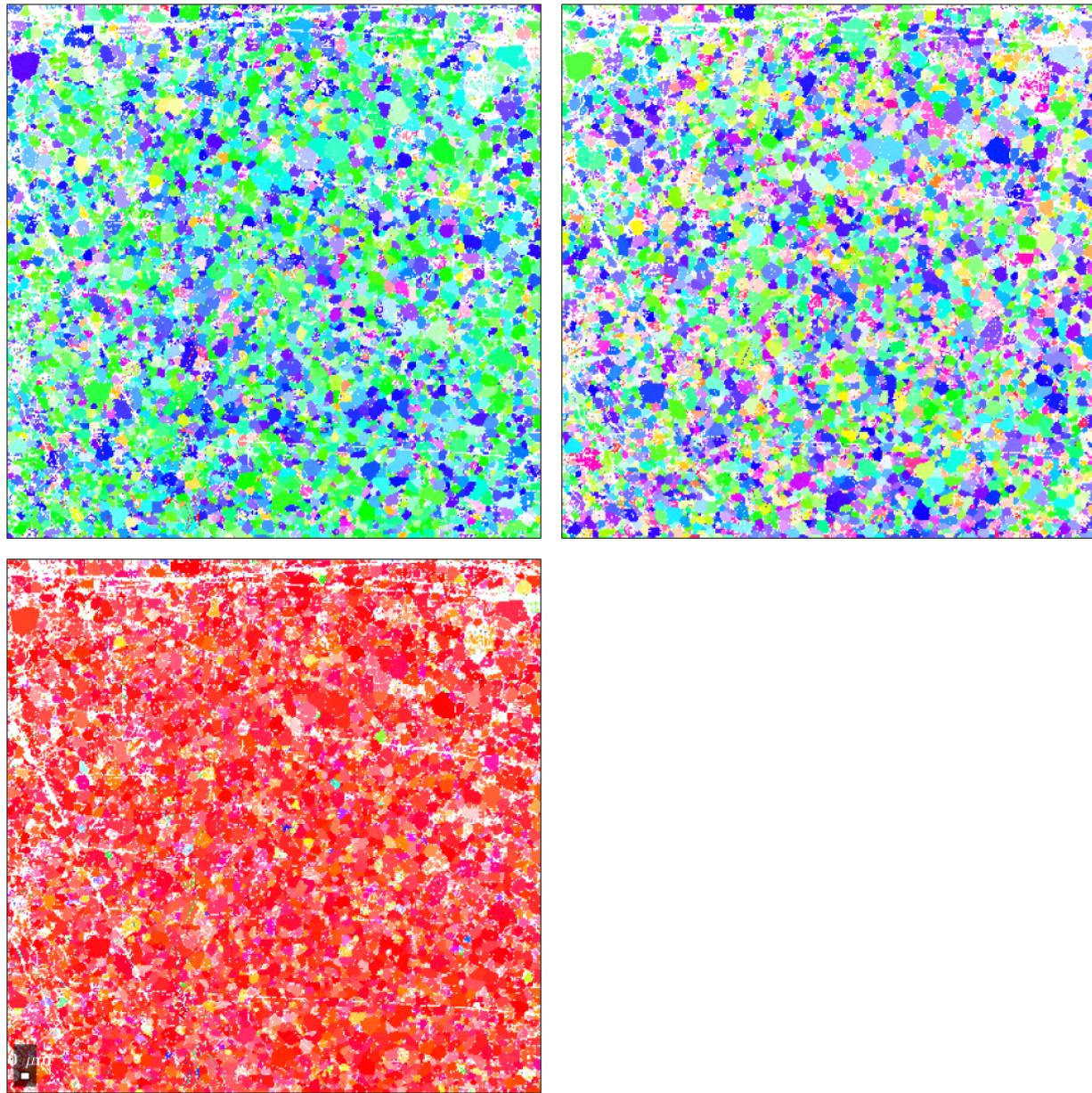
%save the figures;
drawnow(); %updates the graphics engine to make sure it saves things properly
print(f1,fullfile(resultsdir,'Map_IPFkey.png'),'-dpng','-r600');
print(f2,fullfile(resultsdir,'Map_IPF.png'),'-dpng','-r600');
```



6/mmm

[$\bar{1}100$]



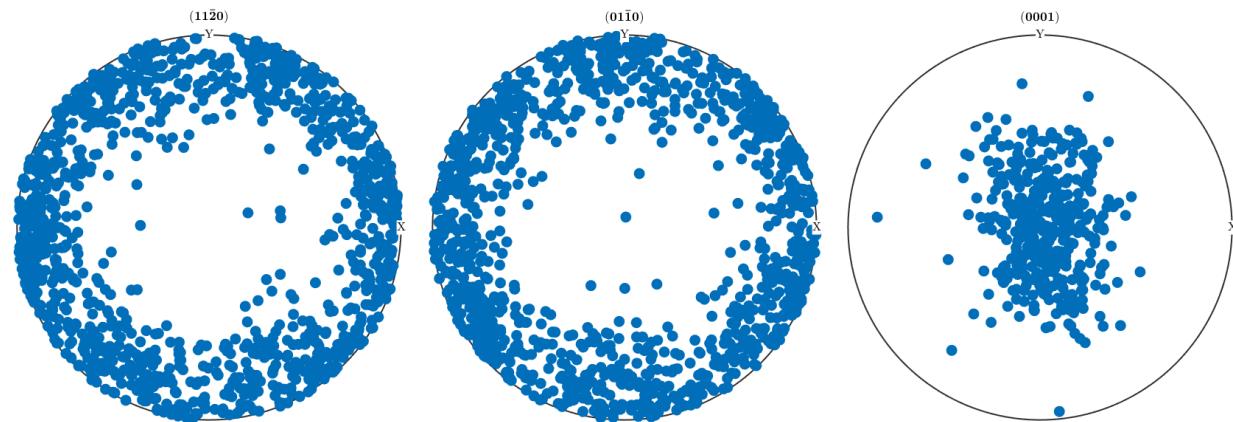


Plot the PF - scatter

```
%plot planes as spots, and use a subset
figure
plotPDF(ebsd(mineral_name).orientations,[Miller(1,1,0,ebsd(mineral_name).CS)
    Miller(0,1,0,ebsd(mineral_name).CS) Miller(0,0,1,ebsd(mineral_name).CS)])
%you can plot all - but this takes a while..
%plotPDF(ebsd(mineral_name).orientations,[Miller(1,1,0,ebsd(mineral_name).CS)
    Miller(0,1,0,ebsd(mineral_name).CS)
    Miller(0,0,1,ebsd(mineral_name).CS)],'Points','all')
```

*I'm plotting 416 random orientations out of 114928 given orientations
 You can specify the the number points by the option "points".*

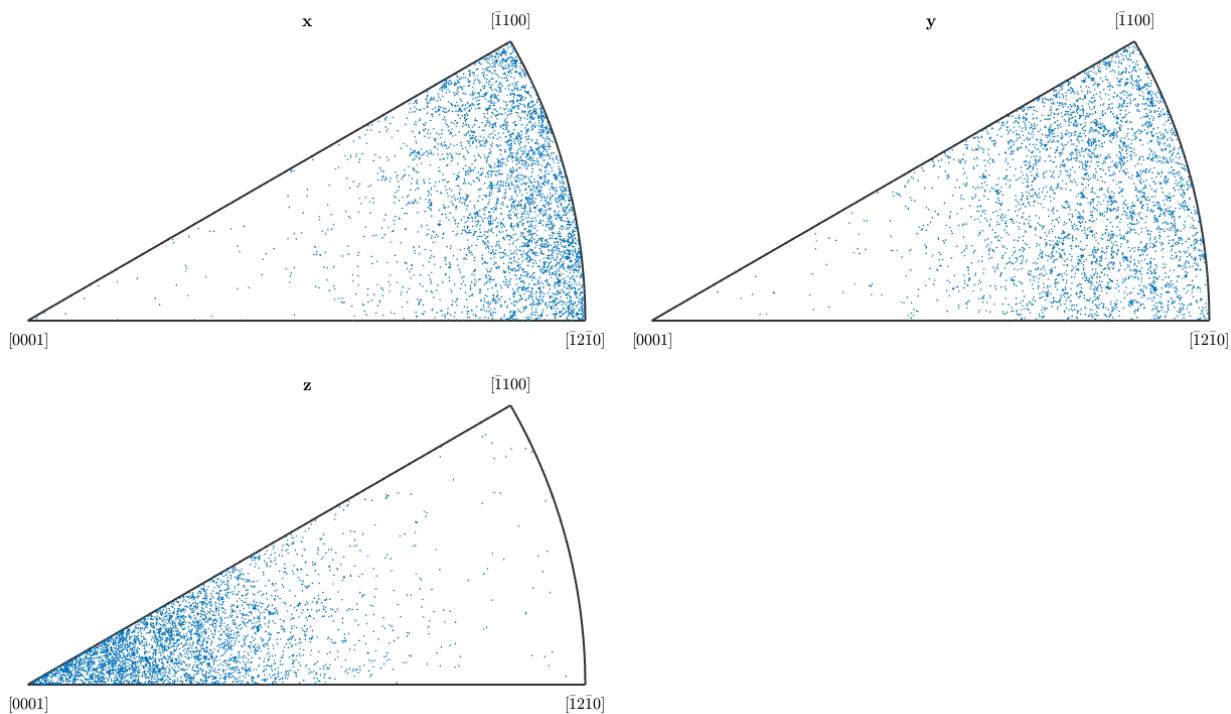
The option "all" ensures that all data are plotted



Plot the IPF - scatter

```
figure  
plotIPDF(ebsd(mineral_name).orientations,[xvector yvector zvector])
```

I'm plotting 4166 random orientations out of 114928 given orientations



Construct the ODF

```
%suggested reading:  
https://mtex-toolbox.github.io/ODFTutorial.html
```

```
%https://mtex-toolbox.github.io/PoleFigure2ODF.html

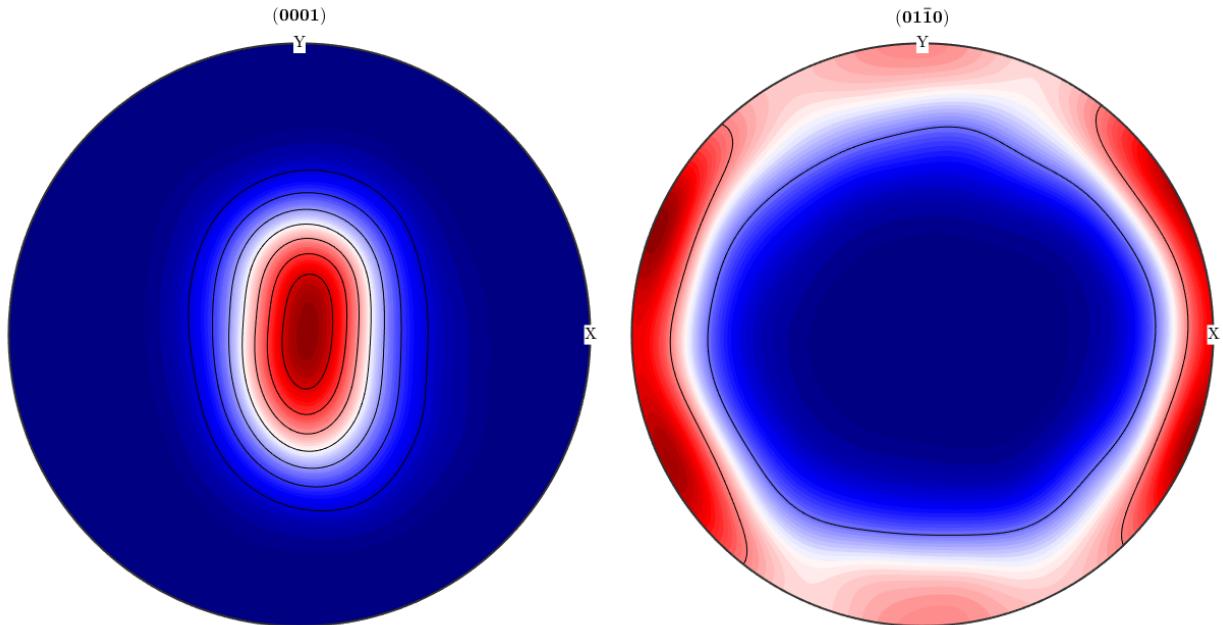
% compute the ODF
odf = calcDensity(ebsd(mineral_name).orientations);
% You could also change/fix the half width - compare this different ODF
% odf = calcDensity(ebsd(mineral_name).orientations,'halfwidth',2*degree);

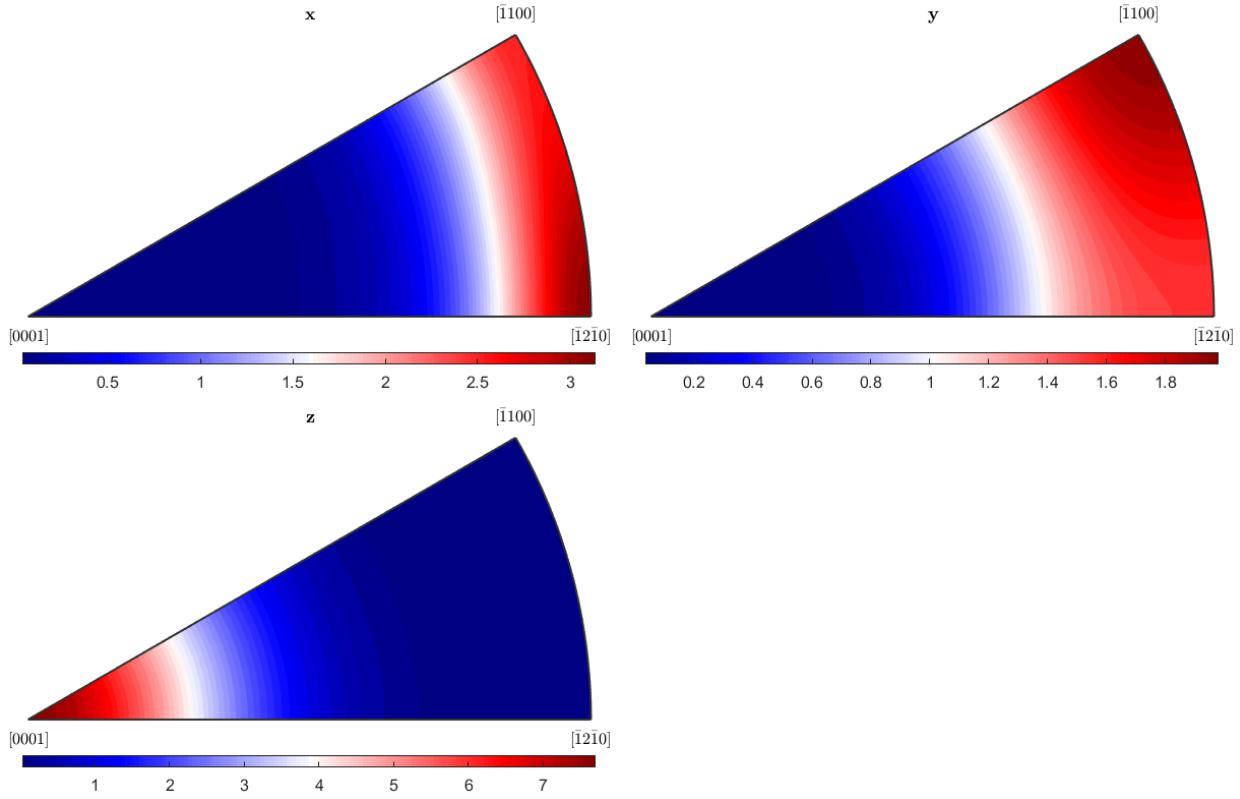
figure;
% plot the pole figure representation of the ODF
plotPDF(odf,[Miller(0,0,1,ebsd(mineral_name).CS)
Miller(0,1,0,ebsd(mineral_name).CS)]);

mtexColorbar('location','southoutside')
mtexColorMap blue2red

%add contours
levels=[-7:7];
hold on;
plotPDF(odf,[Miller(0,0,1,ebsd(mineral_name).CS)
Miller(0,1,0,ebsd(mineral_name).CS)],'contour',levels,'linecolor','k');
mtexColorbar('location','southoutside')

%you can also plot this ODF in the IPF
figure;
% plot the inverse pole figure representation of the ODF
plotIPDF(odf,[xvector yvector zvector])
mtexColorbar('location','southoutside')
mtexColorMap blue2red
```





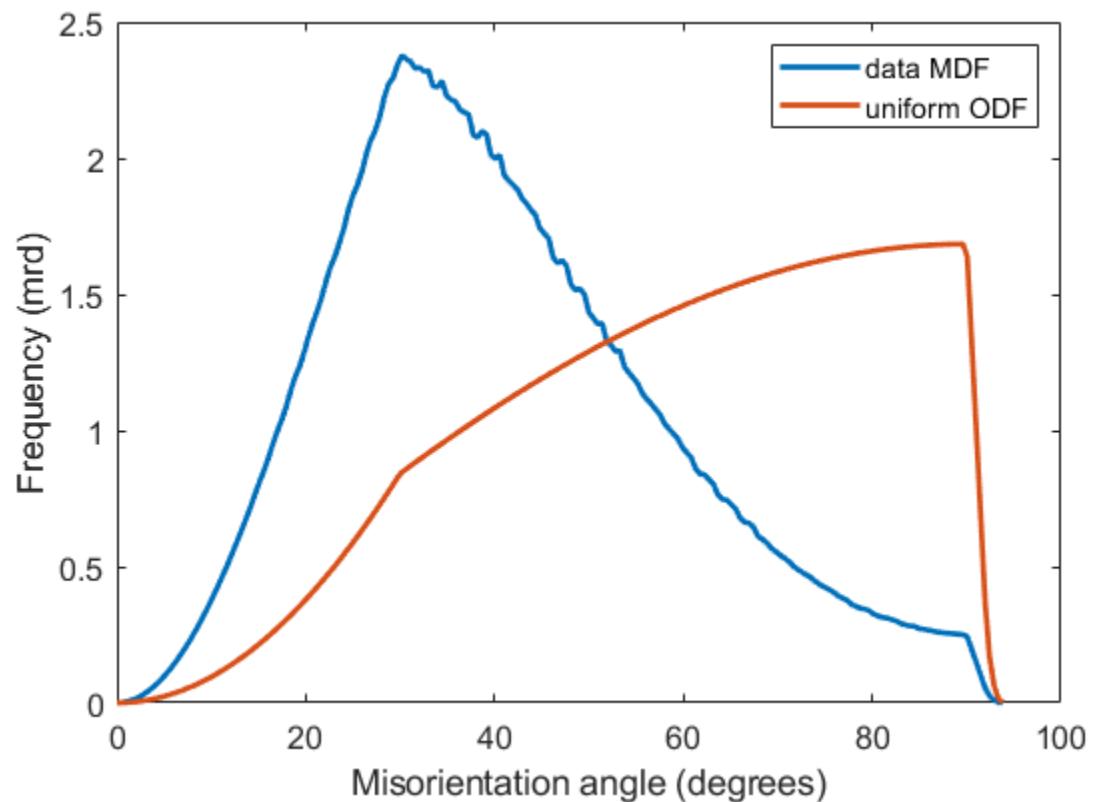
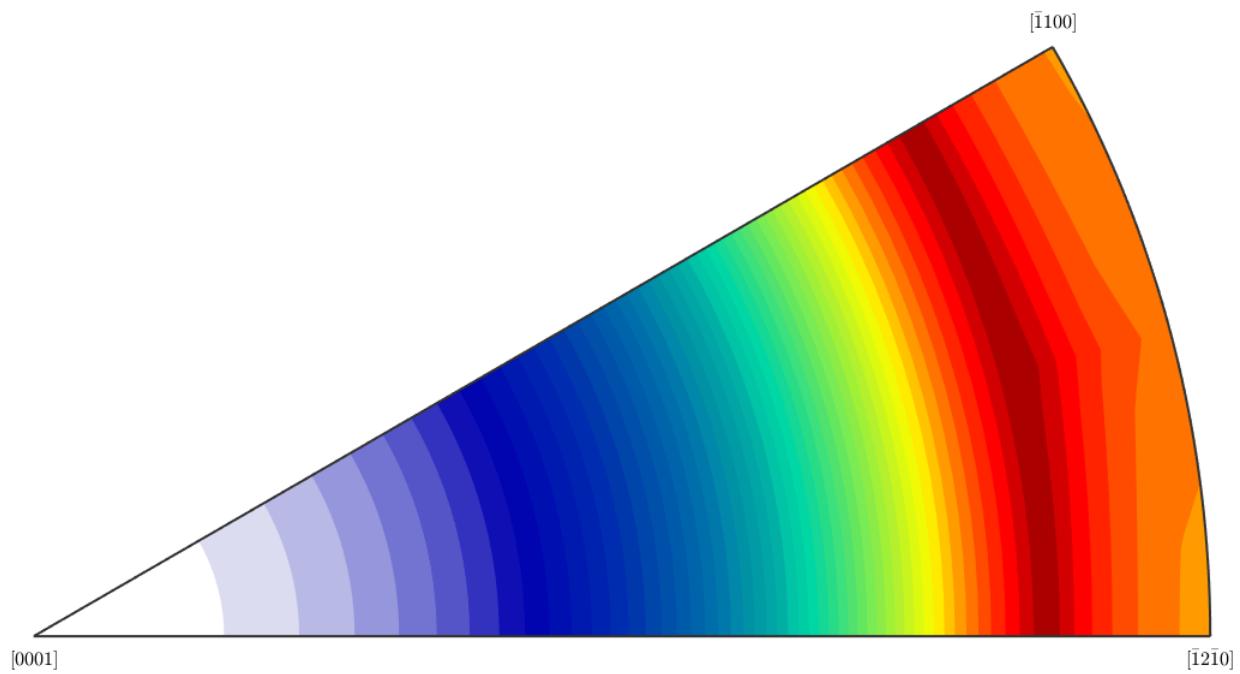
Plot the uncorrelated misorientation distribution

```
%see https://mtex-toolbox.github.io/MisorientationDistributionFunction.html

mdf = calcMDF(odf);

%plot the axis
figure;
plotAxisDistribution(mdf)

%plot the angle
figure;
plotAngleDistribution(mdf)
hold all
plotAngleDistribution(ebsd(mineral_name).CS,ebsd(mineral_name).CS)
hold off
legend('data MDF','uniform ODF')
```



Calculate grains and grain boundaries

%see <https://mtex-toolbox.github.io/GrainReconstruction.html>

```

[grains,ebsd.grainId] = calcGrains(ebsd,'angle',10*degree);
% [grains,ebsd.grainId] =
calcGrains(ebsd,'angle',10*degree,'boundary','tight');

%care with this threshold value - you can look at the angle distribution for
some guidance

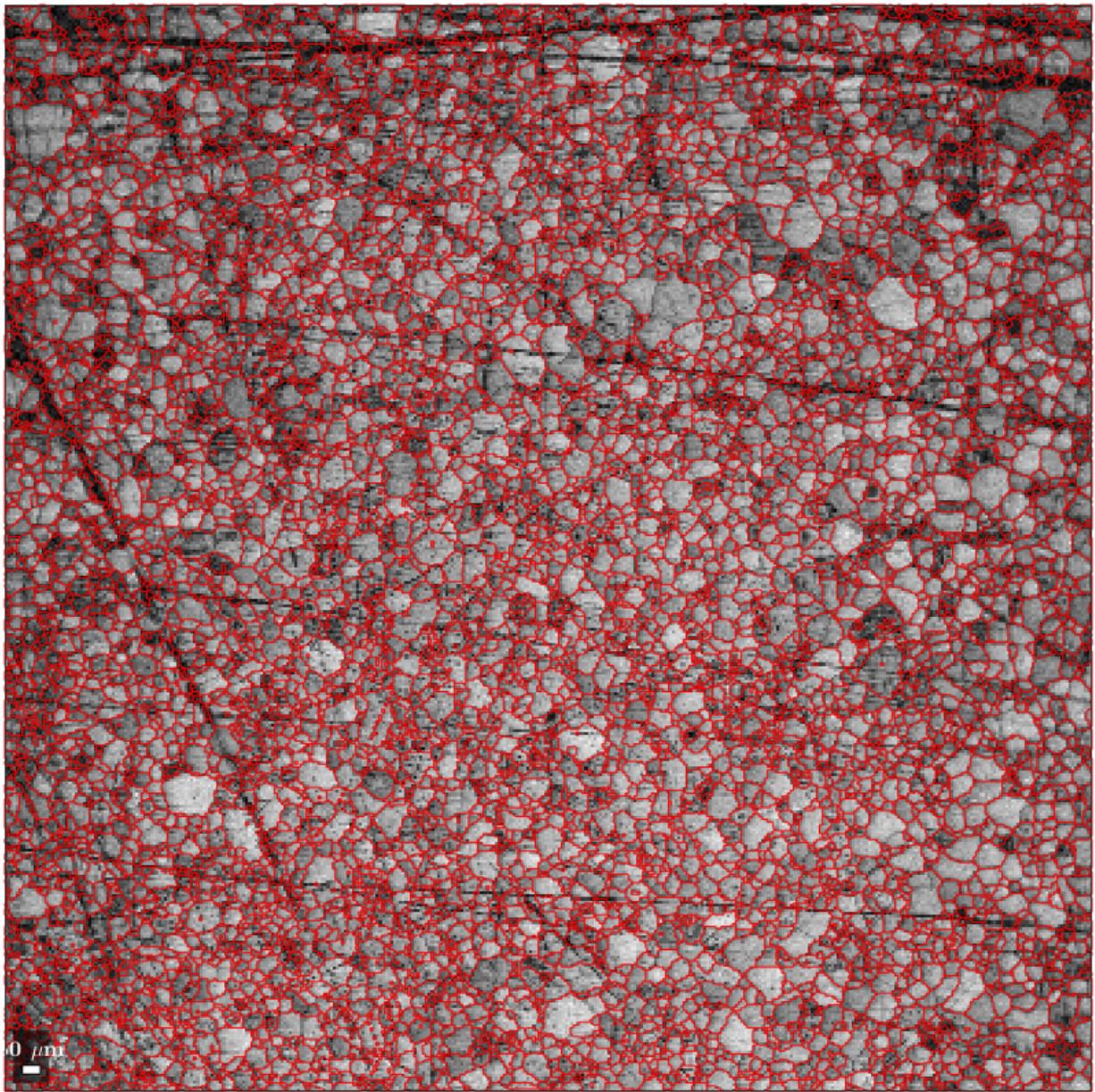
%Now plot the grain boundary map on the band contrast map
figure;
plot(ebsd,ebsd.prop.Band_Contrast); colormap('gray');
if exist('mb_length','var'); mp = MTEX_mb_fix(mb_length); end %change the
micronbar length to make it prettier/easier to read
hold on
plot(grains.boundary,'linewidth',0.5,'lineColor','r')
hold off

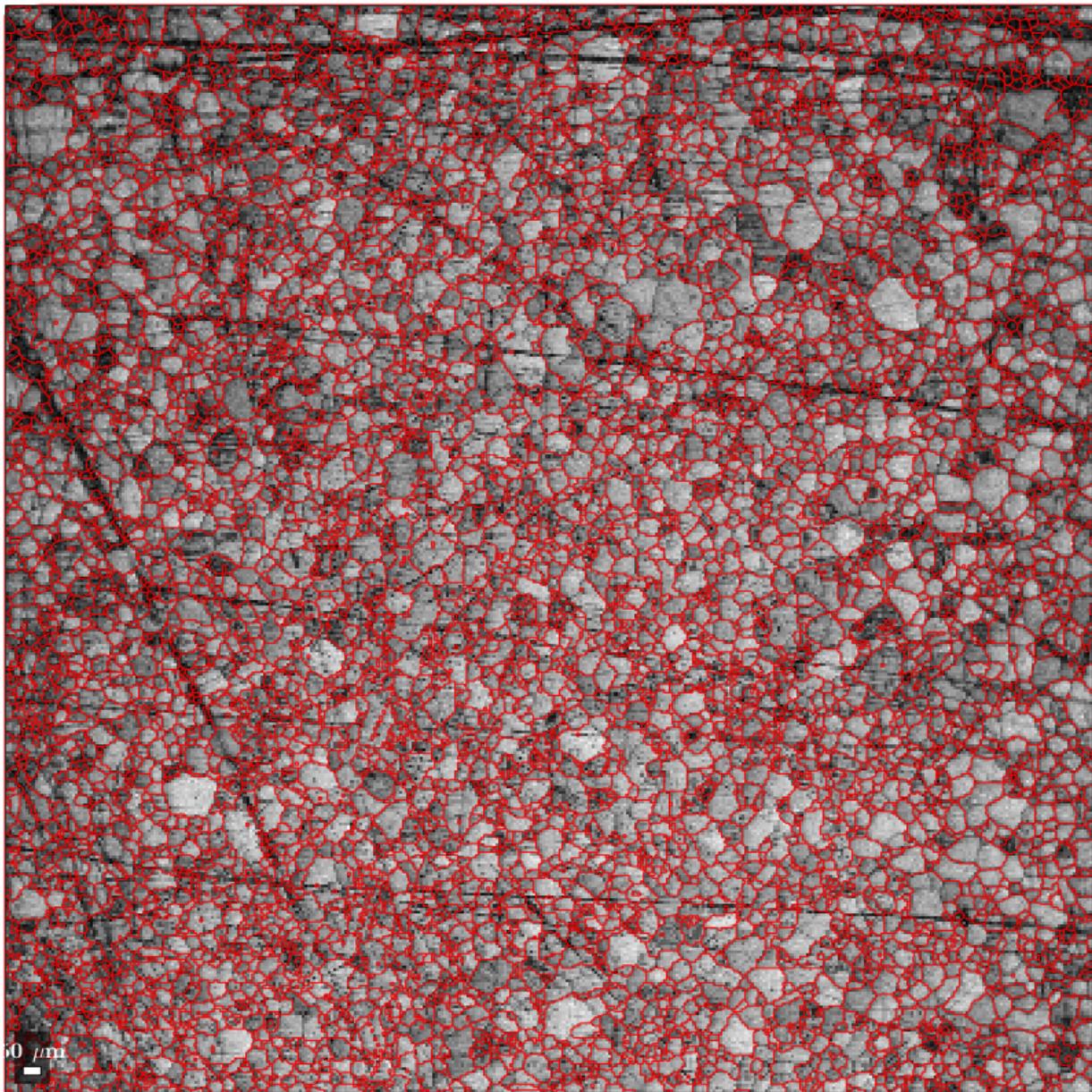
%contrast this with the grain boundaries for indexed only points
%(look at the scratches)
grains_indexed = calcGrains(ebsd('indexed'));
% If the edge grains are being annoying - don't use the convex hull
% grains_indexed = calcGrains(ebsd('indexed'),'boundary','tight');

figure;
plot(ebsd,ebsd.prop.Band_Contrast); colormap('gray');
if exist('mb_length','var'); mp = MTEX_mb_fix(mb_length); end %change the
micronbar length to make it prettier/easier to read
hold on
plot(grains_indexed.boundary,'linewidth',0.5,'lineColor','r')
hold off

%subset to only have the interior grains
grains_indexed_int=grains_indexed(~grains_indexed.isBoundary);
% hold on
% plot(grains_indexed_int.boundary,'linewidth',0.5,'lineColor','g')
% hold off

```





Plot the grain size distributions

```
%extract some different distributions
grain_radius=grains_indexed_int.equivalentRadius;
grain_area=grains_indexed_int.area;
grain_perimeter=grains_indexed_int.equivalentPerimeter;
% for a full list of shapes - https://mtex-toolbox.github.io/ShapeParameters.html

histogram_bins=100;

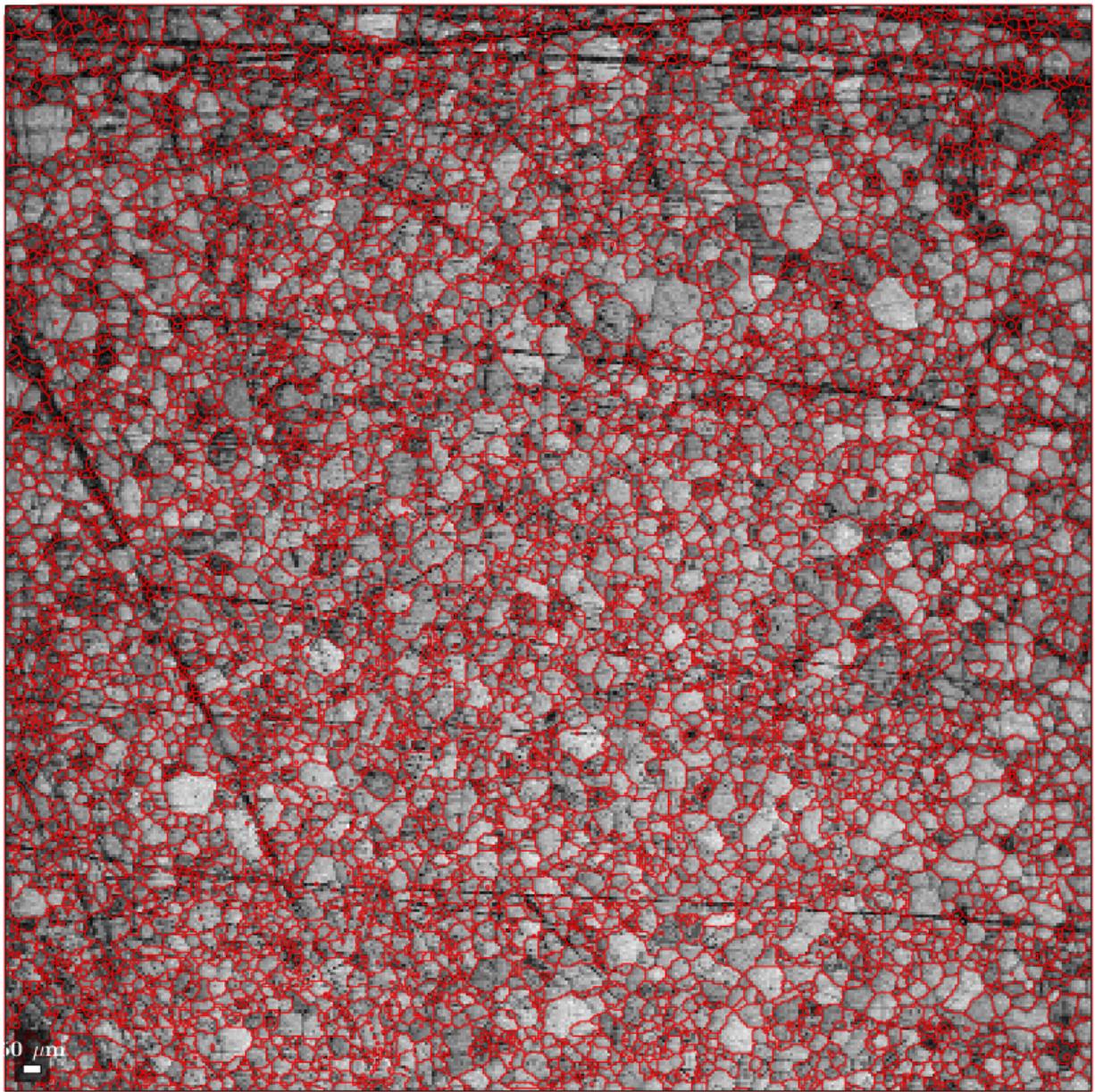
figure;
subplot(3,1,1);
```

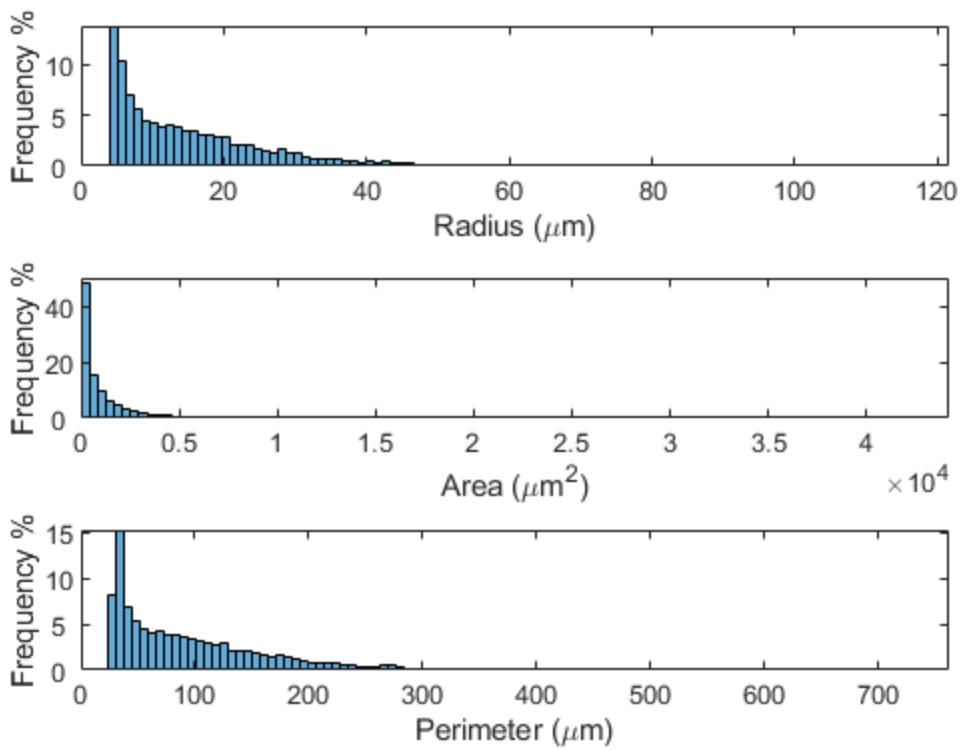
```
histogram(grain_radius,histogram_bins,'Normalization','probability');
%meddle with the displays
ytix = get(gca, 'YTick');
set(gca, 'YTick',ytix, 'YTickLabel',ytix*100);
ylabel('Frequency %'); xlabel('Radius (\mu m)')
xlims=xlim; xlim([0 xlims(2)]);

subplot(3,1,2);
histogram(grain_area,histogram_bins,'Normalization','probability');
%meddle with the displays
ytix = get(gca, 'YTick');
set(gca, 'YTick',ytix, 'YTickLabel',ytix*100);
ylabel('Frequency %'); xlabel('Area (\mu m^2)')
xlims=xlim; xlim([0 xlims(2)]);

subplot(3,1,3);
histogram(grain_perimeter,histogram_bins,'Normalization','probability');
%meddle with the displays
ytix = get(gca, 'YTick');
set(gca, 'YTick',ytix, 'YTickLabel',ytix*100);
ylabel('Frequency %'); xlabel('Perimeter (\mu m)')
xlims=xlim; xlim([0 xlims(2)]);

%are any of these Normally distributed ? If so, have a think about what you
%call an average 'grain size'
```





Filter the grain size data

%mostly to demo that this can be done

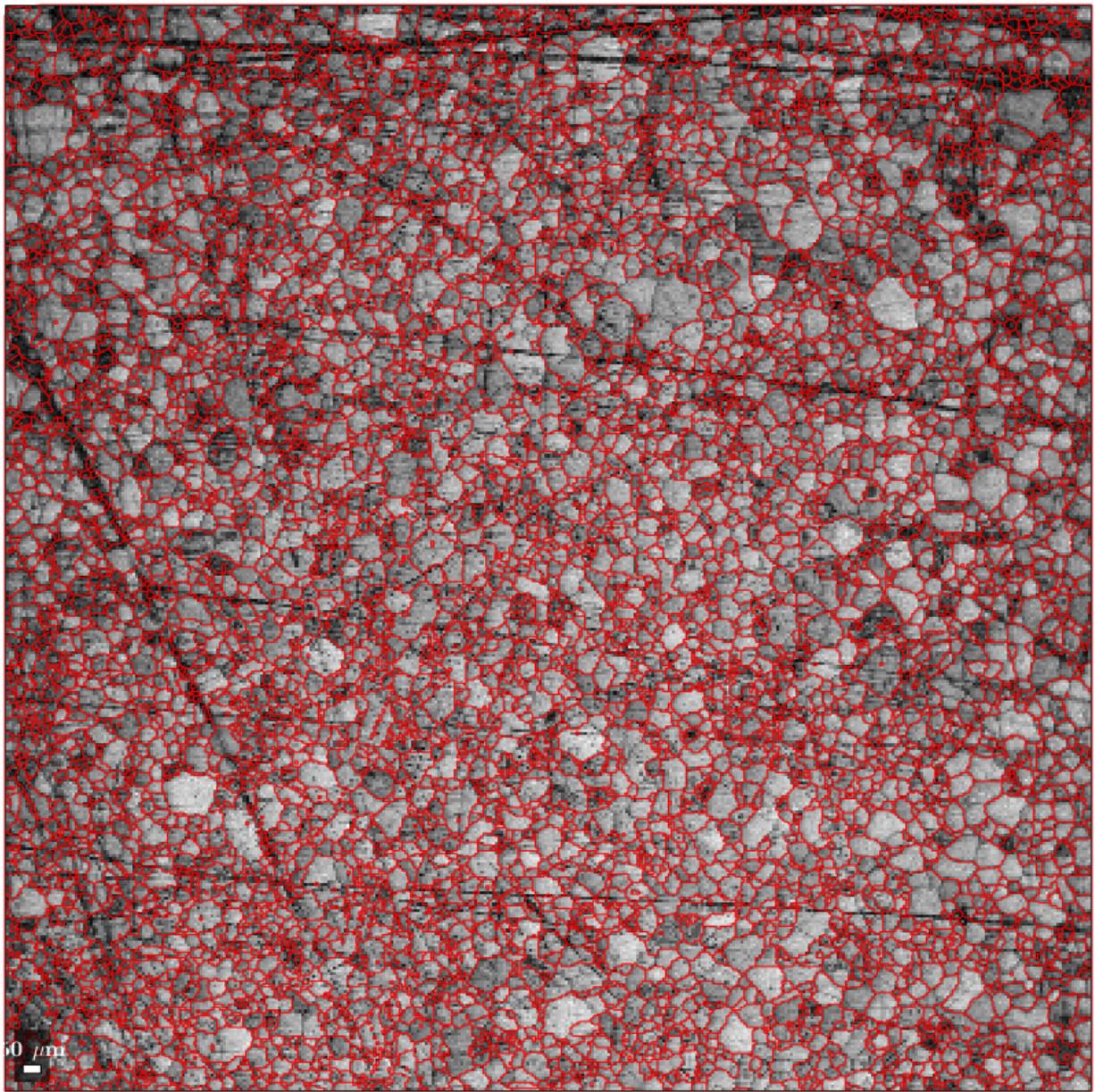
```

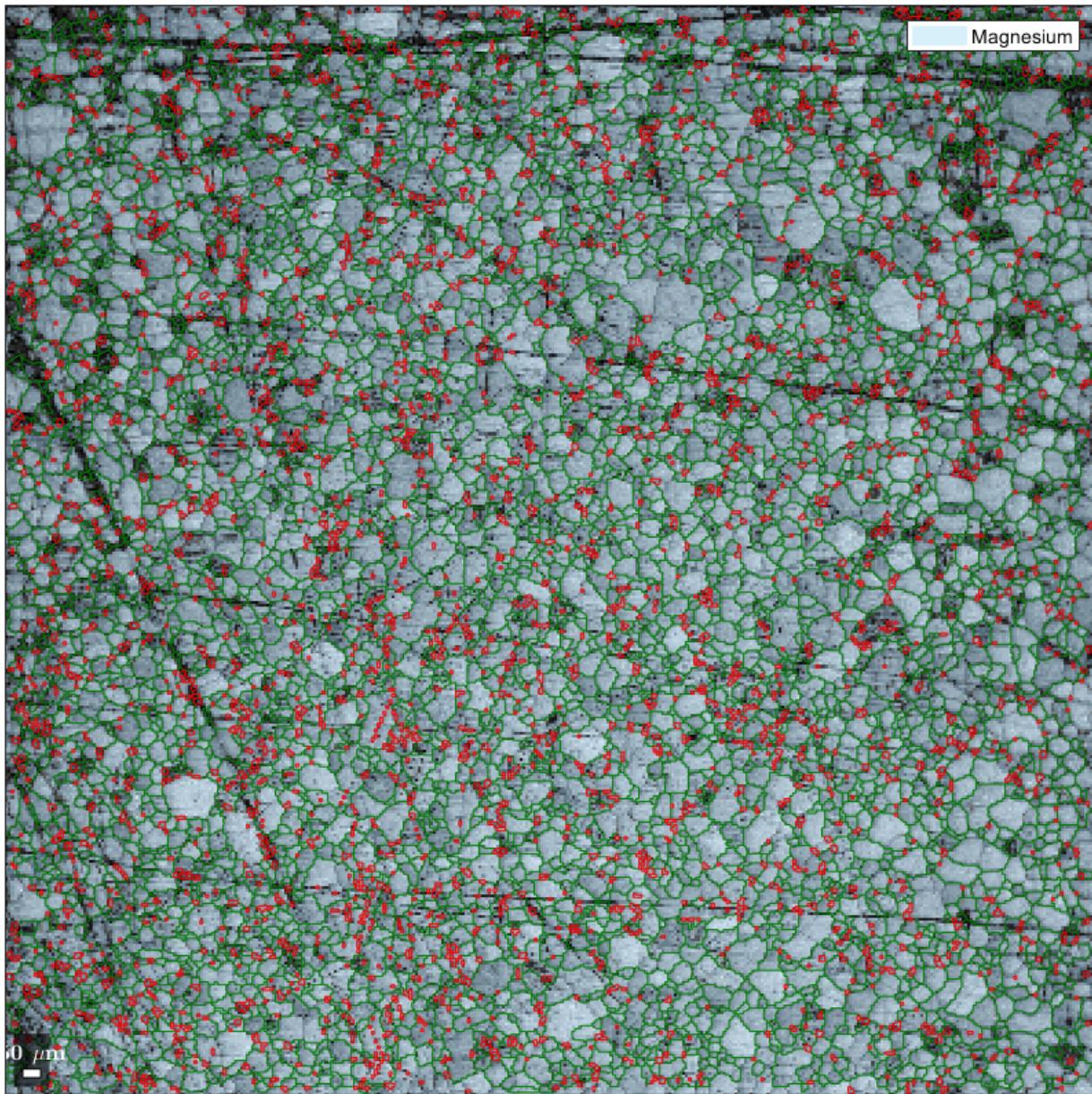
min_grain_radius=10;
grains_reduced=grains_indexed_int(grains_indexed_int.equivalentRadius>min_grain_radius);
grains_excluded=grains_indexed_int(grains_indexed_int.equivalentRadius<=min_grain_radius);
figure;
plot(ebsd,ebsd.prop.Band_Contrast); colormap('gray');
hold on;
plot(ebsd,'FaceAlpha','0.3');
hold on;
if exist('mb_length','var'); mp = MTEX_mb_fix(mb_length); end %change the
micronbar length to make it prettier/easier to read

hold on
plot(grains_reduced.boundary,'LineWidth',0.5,'LineColor','g')
plot(grains_excluded.boundary,'LineWidth',0.5,'LineColor','r')

hold off

```





Demonstrate that we can extract a point from this map

```
%plot the phase data + grain boundaries + bc
figure;
plot(ebsd');

%overlay with Band Contrast
figure;
plot(ebsd,ebsd.prop.Band_Contrast); colormap('gray');
hold on;
plot(ebsd,'FaceAlpha','0.3');
```

```

hold on;
if exist('mb_length','var'); mp = MTEX_mb_fix(mb_length); end %change the
micronbar length to make it prettier/easier to read

hold on
plot(grains_indexed_int.boundary,'linewidth',0.5,'lineColor','g')
hold off

%take a mouse input
disp('Click a grain to select it')
[x_g,y_g]=ginput(1);

%find the nearest point in the data to this point
p_map=(ebsd.prop.x-x_g).^2+(ebsd.prop.y-y_g).^2;
[mv,pattern_number]=min(reshape(p_map,[],1));
ebsd_point=ebsd(pattern_number);

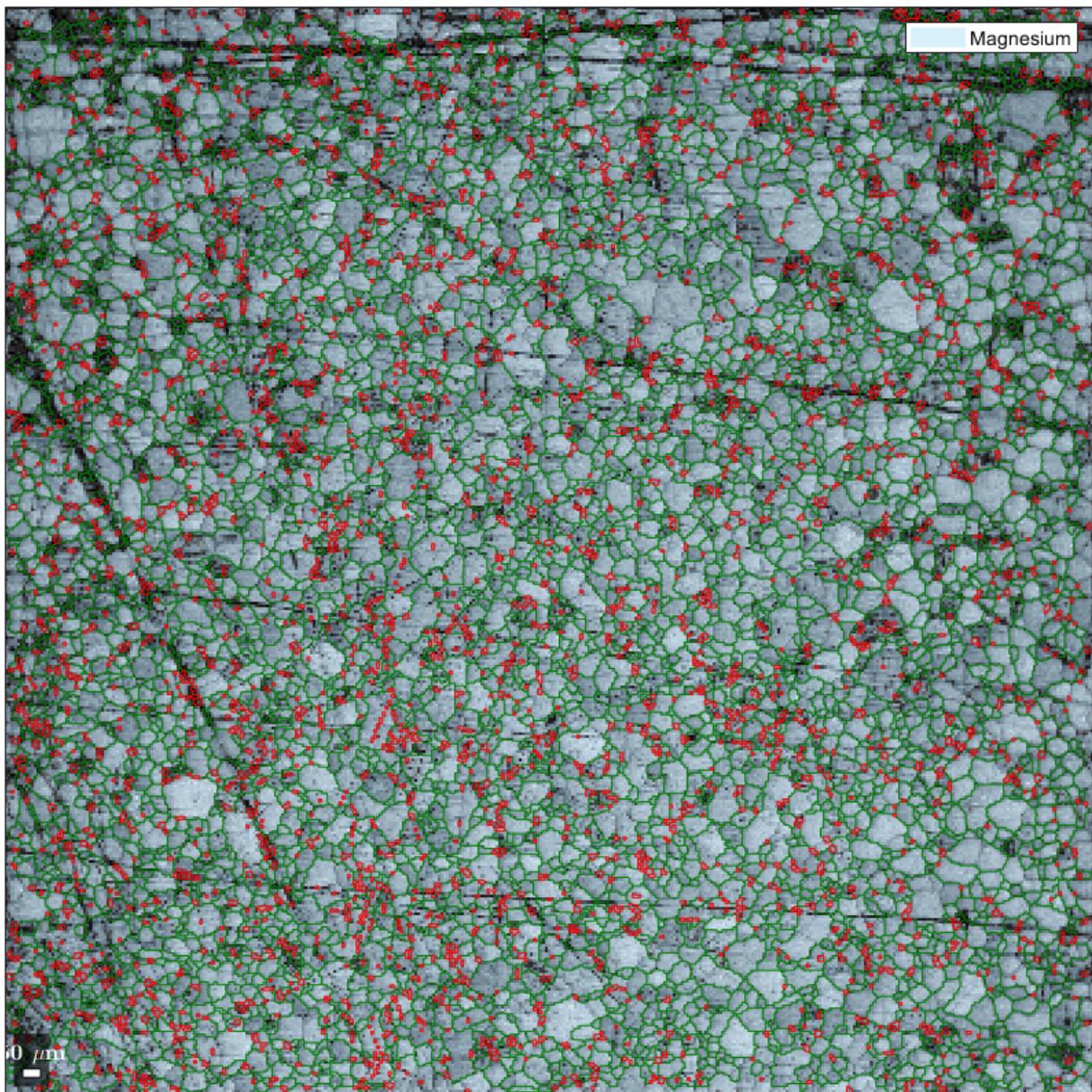
hold on;
% % scatter(x_g,y_g,'r');
%mark this point
scatter(ebsd_point.x,ebsd_point.y,'w');

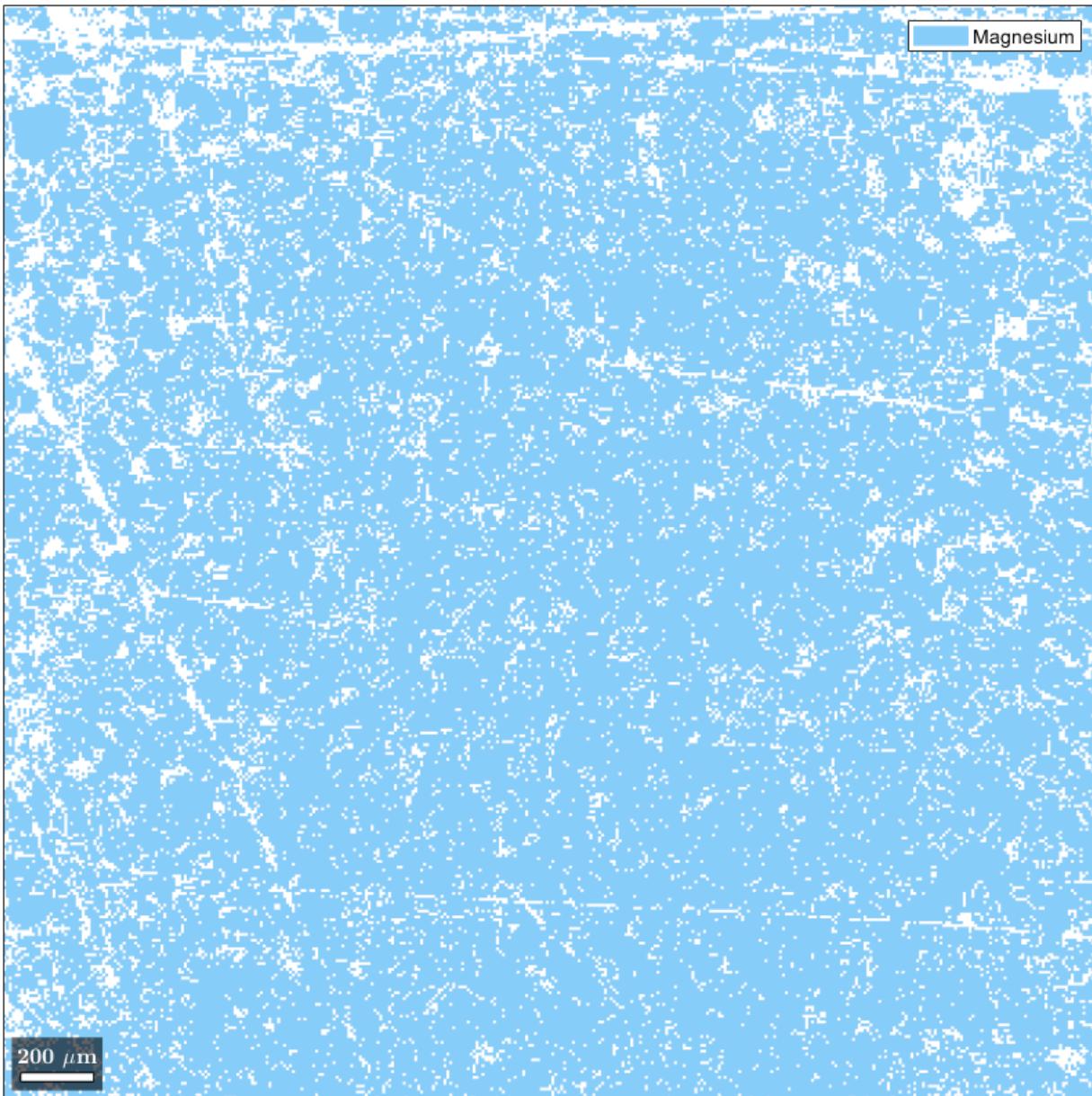
%plot the grain for this
hold on
plot(grains(ebsd_point.x,ebsd_point.y).boundary,'linewidth',4,'linecolor','y')
hold off

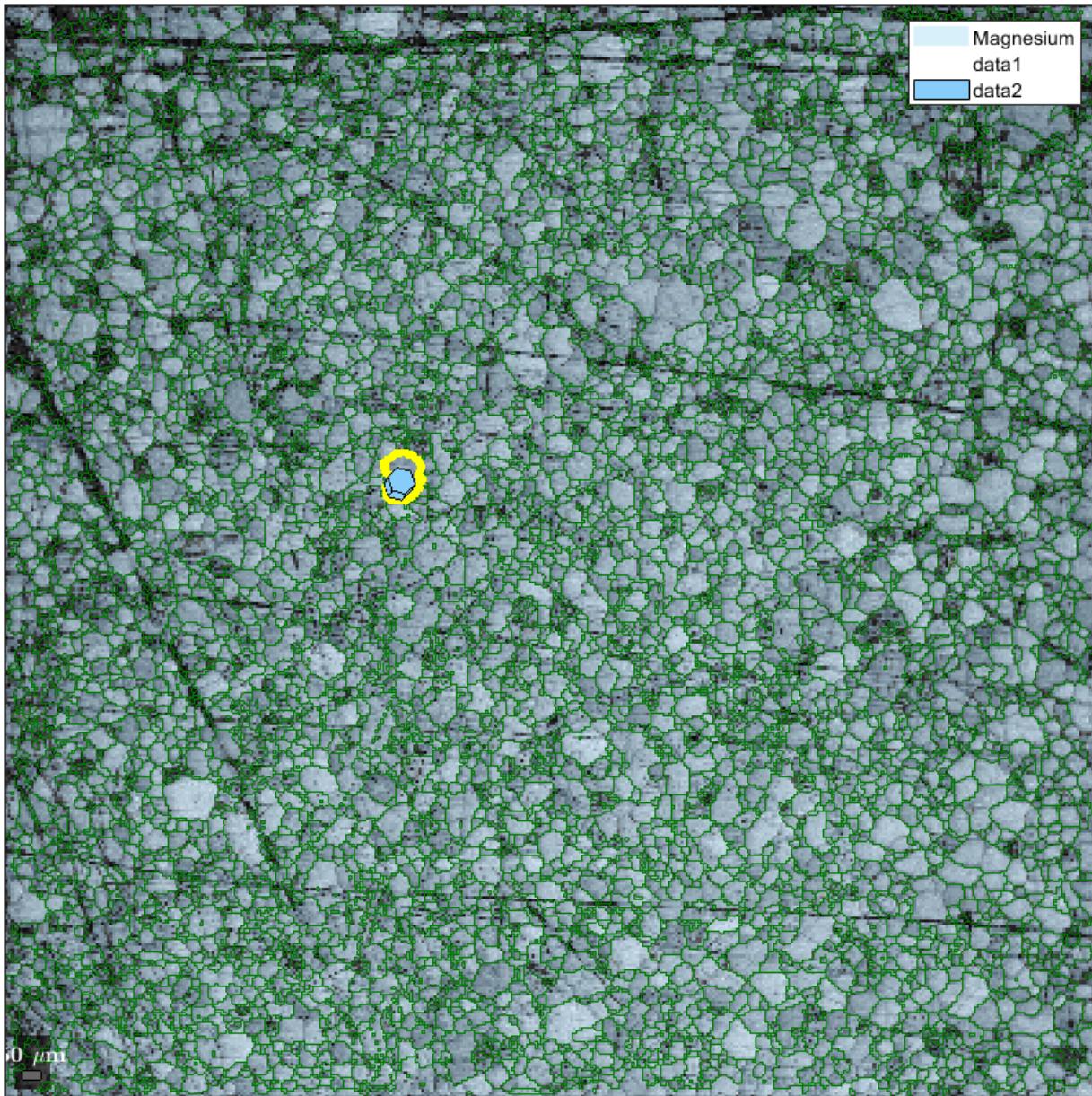
%plot the crystal shape for this point
hold on
scaling = 100; % scale the crystal shape to have a nice size
%here we use a hexagon, but you could use a cube for cubic
%crystalShape.cube
cS = crystalShape.hex(ebsd(mineral_name).CS);
plot(ebsd(pattern_number).prop.x,ebsd(pattern_number).prop.y,50,
    ebsd(pattern_number).orientations * cS * scaling)
hold off

Click a grain to select it

```







Extract a subset of the map

```
figure;
plot(ebsd,ebsd.prop.Band_Contrast)

%here we will use a mouse click pair to cut out the box - pick two opposite
%diagonals
disp('Click two diagonal points to select a box to subset the data')
[x_i,y_i]=ginput(2);

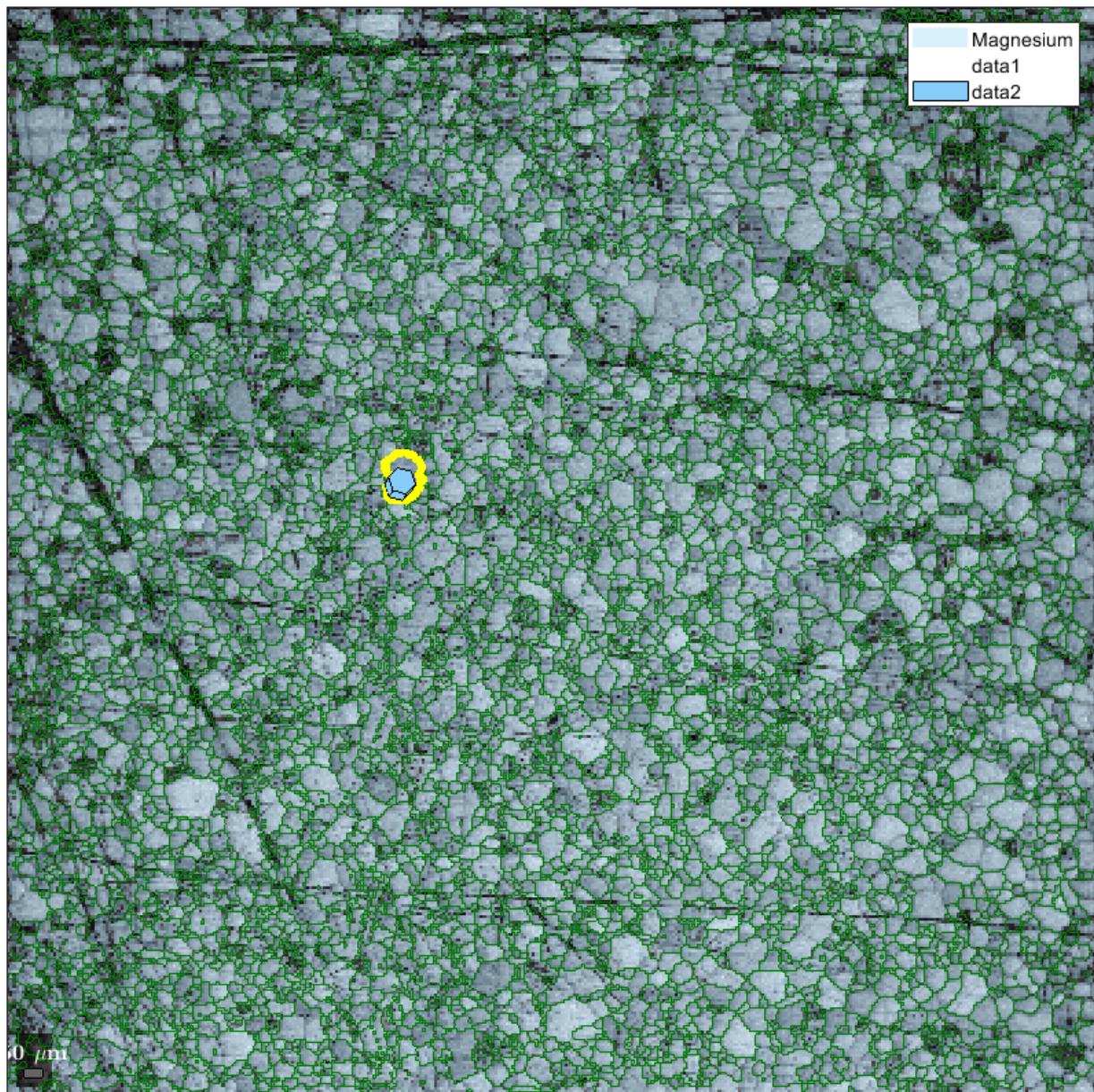
% x_i=[-918;-205];
% y_i=[572;1010];
```

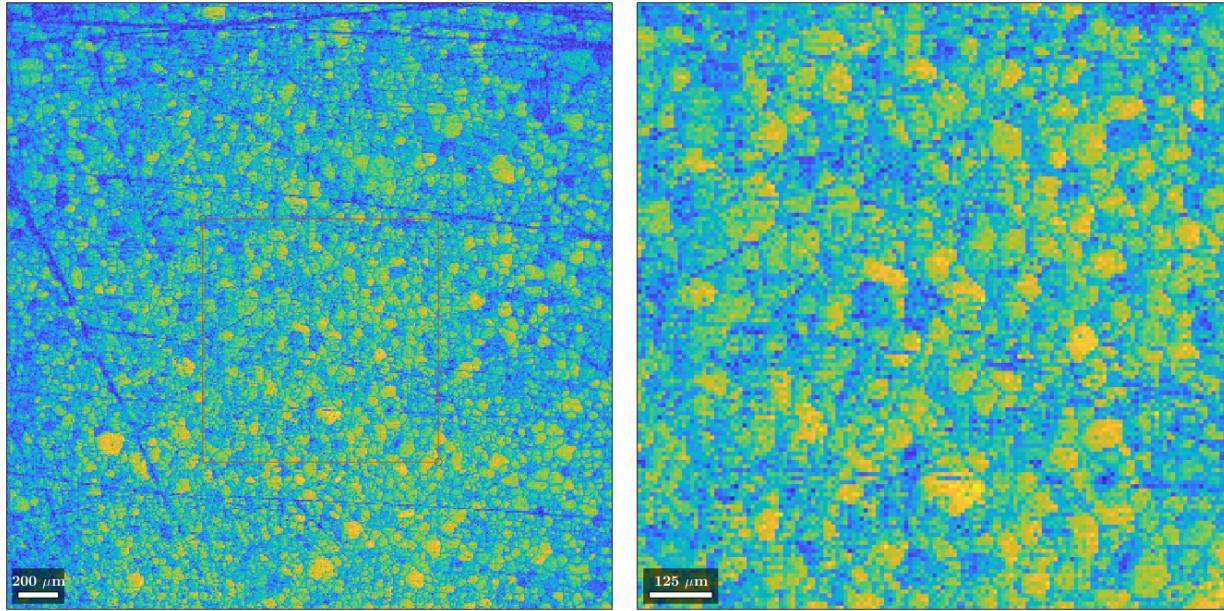
```
hold on;
scatter(x_i,y_i);

x_i=sort(x_i); y_i=sort(y_i);
box_bounds=[x_i(1),x_i(2),x_i(2),x_i(1),x_i(1);y_i(1),y_i(1),y_i(2),y_i(2),y_i(1)];
plot(box_bounds(1,:),box_bounds(2,:));
points_box=inpolygon(ebsd,box_bounds');
ebsd_small=ebsd(points_box);

nextAxis;
plot(ebsd_small,ebsd_small.prop.Band_Contrast);
```

Click two diagonal points to select a box to subset the data





For this smaller region, plot all the IPFs

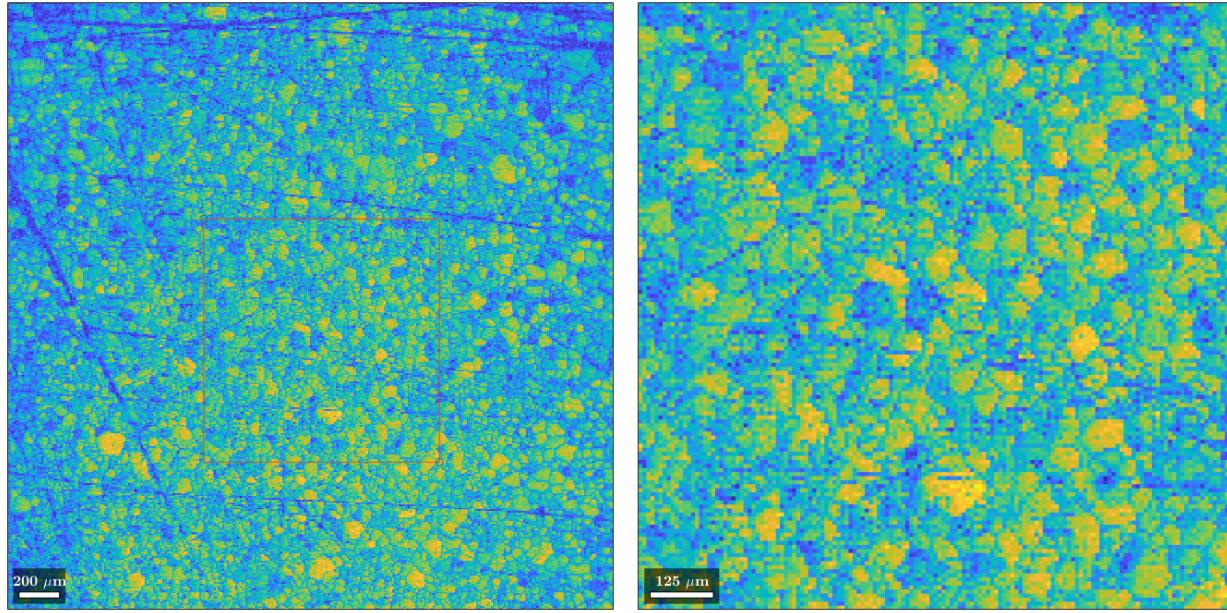
```

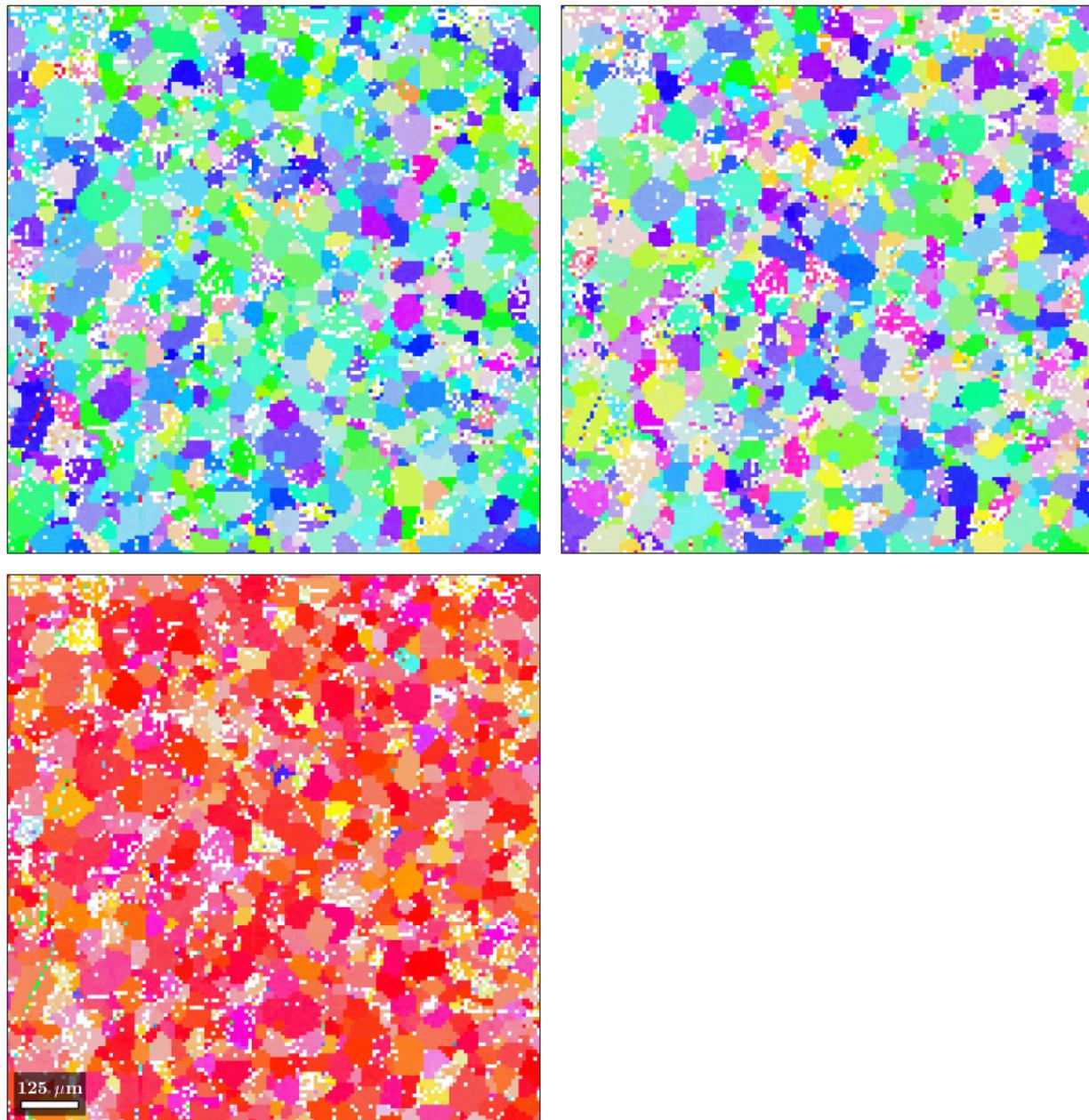
ipfKey2 = ipfHSVKey(ebsd_small(mineral_name));

ipfKey2.inversePoleFigureDirection = vector3d.X;
colors_X2 = ipfKey2.orientation2color(ebsd_small(mineral_name).orientations);
ipfKey2.inversePoleFigureDirection = vector3d.Y;
colors_Y2 = ipfKey2.orientation2color(ebsd_small(mineral_name).orientations);
ipfKey2.inversePoleFigureDirection = vector3d.Z;
colors_Z2 = ipfKey2.orientation2color(ebsd_small(mineral_name).orientations);

%now plot the three IPF coloured maps
figure;
plot(ebsd_small(mineral_name),colors_X2,'micronbar','off');
nextAxis
plot(ebsd_small(mineral_name),colors_Y2,'micronbar','off');
nextAxis
plot(ebsd_small(mineral_name),colors_Z2,'micronbar','on');

```





**extract the grain data from the big map that fits
within this smaller map**

```
grains_small_ID=inpolygon(grains,box_bounds');
grains_small=grains(grains_small_ID);
grains_small=grains_small(grains_small.grainSize>10); %at least 10 pts per
grain

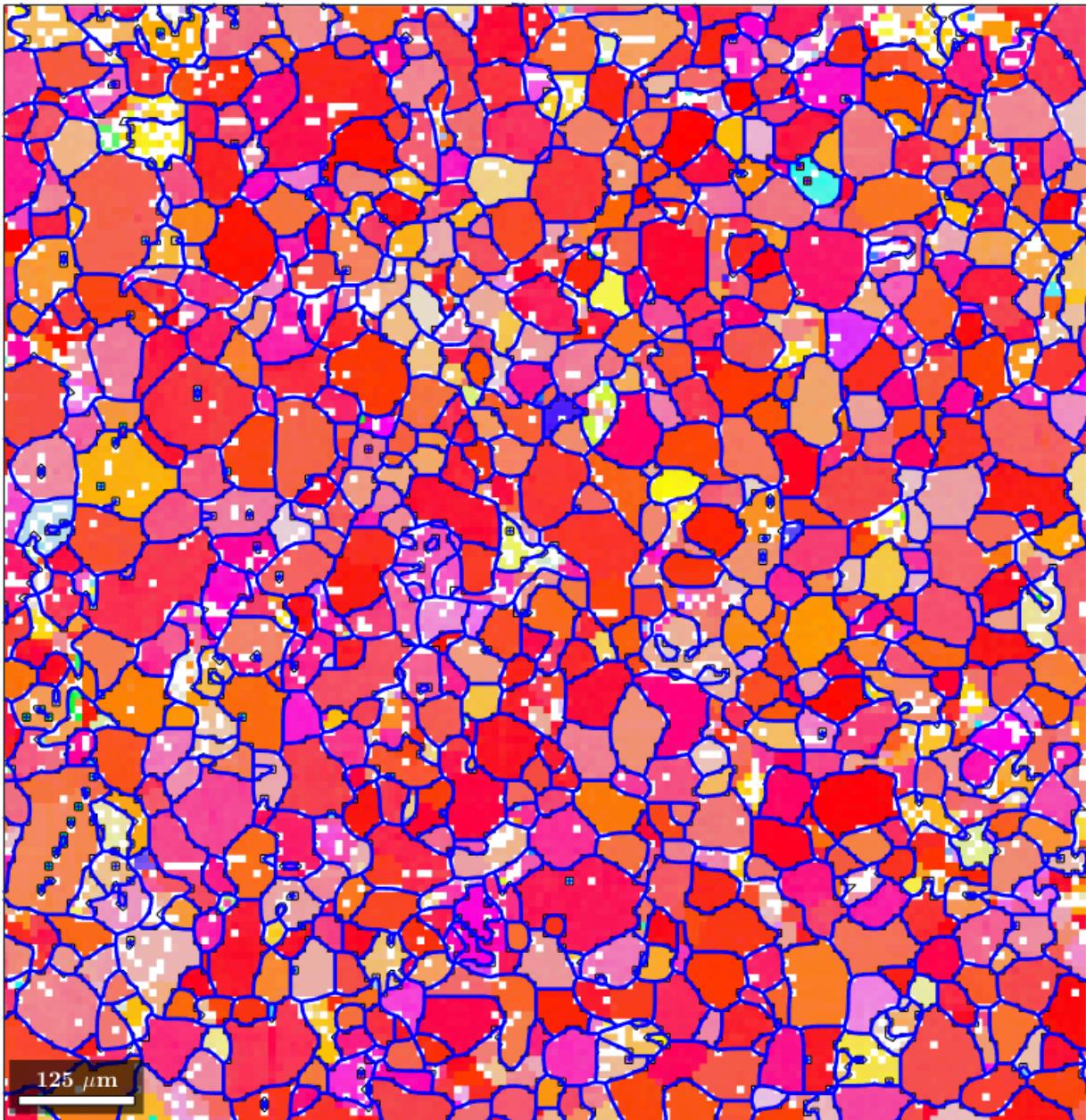
figure;
plot(ebsd_small(mineral_name),colors_Z2,'micronbar','on');
```

```
hold on
plot(grains_small.boundary,'LineWidth',0.5,'LineColor','k')
hold off

%smooth these boundaries - note smoothing changes the data and its statistics
% see - https://mtex-toolbox.github.io/GrainSmoothing.html
% grains_smooth = smooth(grains_small); %single pass
grains_smooth = smooth(grains_small,5); %5 iterations
% see https://mtex-toolbox.github.io/GrainSmoothing.html

hold on
plot(grains_smooth.boundary,'LineWidth',2,'LineColor','b')
hold off

%fix the bounding edges
xlim(x_i); ylim(y_i);
```



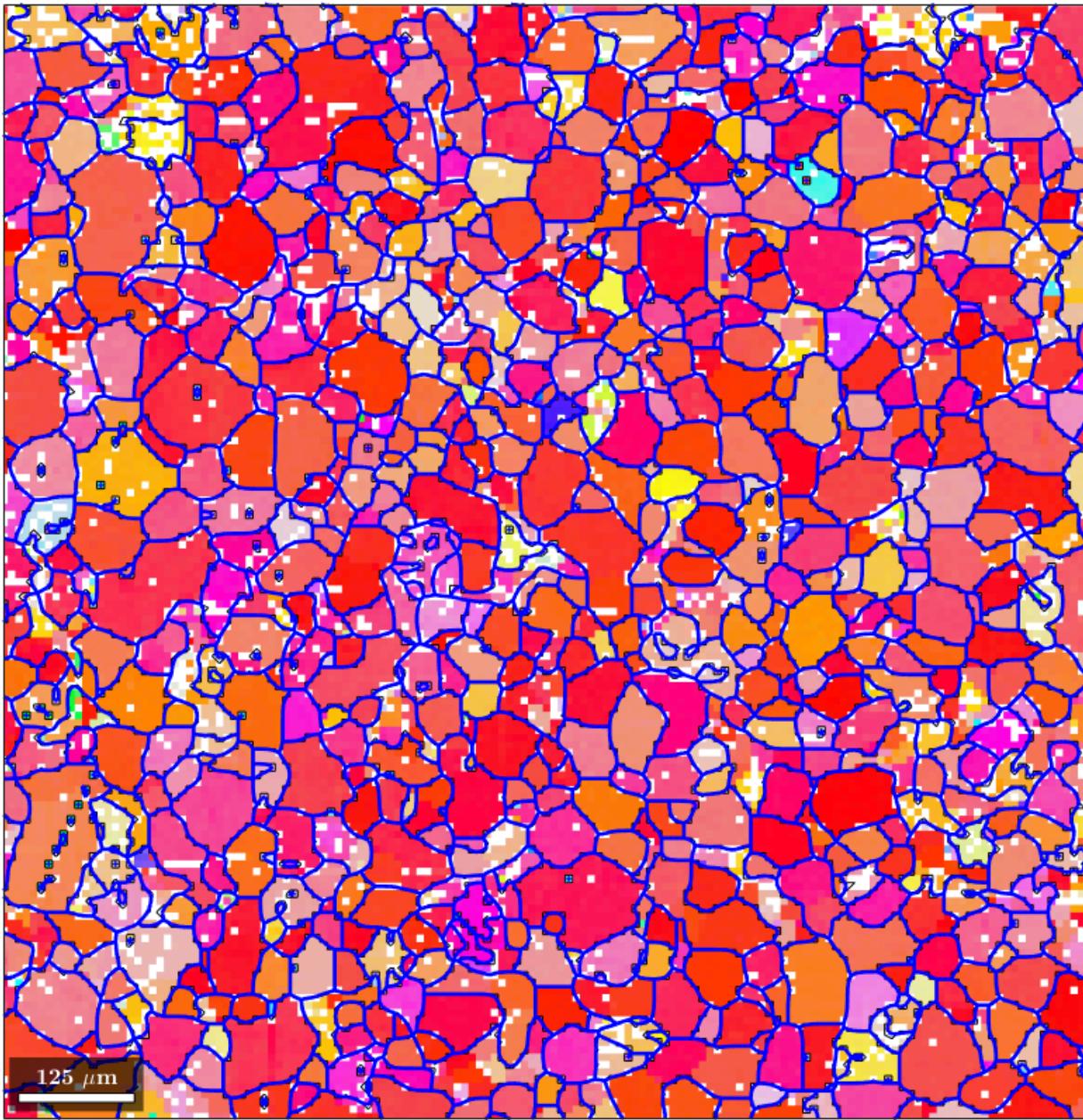
Plot a subset map with the unit cells over the top

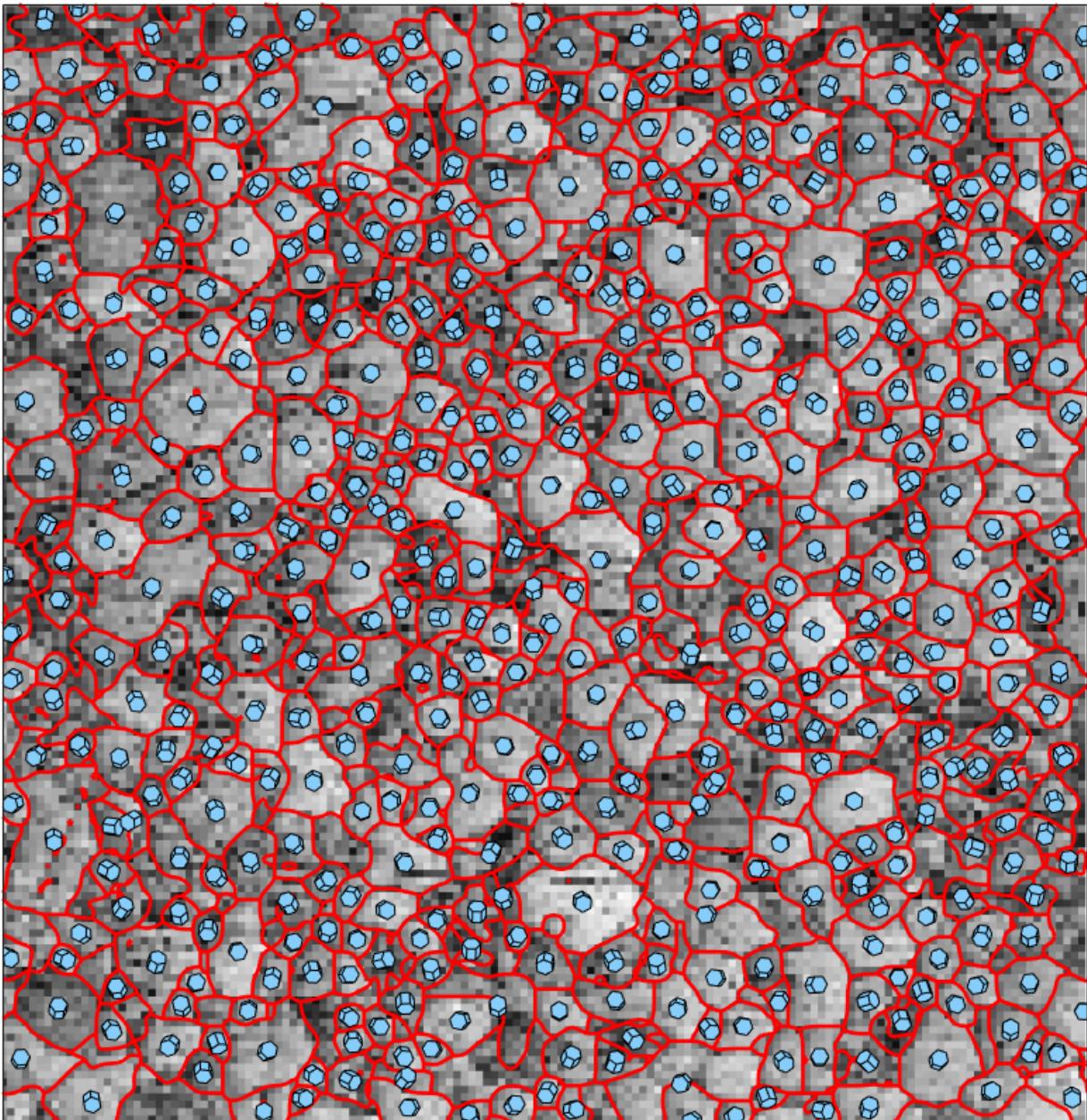
```
figure;
plot(ebsd_small,ebsd_small.prop.Band_Contrast,'micronbar','off'); %the scale
bar is turned off because we are going to make it hard to see
colormap('gray');
hold on
plot(grains_smooth.boundary,'linewidth',2,'lineColor','r')

% plot the unit cells
```

```
%plot the crystal shape for this point
scaling = 25; % scale the crystal shape to have a nice size
%here we use a hexagon, but you could use a cube for cubic
%crystalShape.cube
cS = crystalShape.hex(grains_small(mineral_name).CS);
grain_centroids=grains_small.centroid;
plot(grain_centroids(:,1),grain_centroids(:,2),50,
    grains_small.meanOrientation * cS * scaling)

%fix the bounding edges
xlim(x_i); ylim(y_i);
```





Select an individual grain for texture based segmentation

```
use grains(x,y) to select a grain from a map based on x and y coordinates grain_selected = grains(200,500);
```

```
%3 check you have the right grain by plotting, then calculate the mean  
%orientation
```

```
%create a container for EBSD phase 1 (alpha - HCP)  
ebsd_2=ebsd(mineral_name);
```

```

% figure('Position',PlotData.ssize);
figure;
subplot(1,3,1);
ipfKey.inversePoleFigureDirection = vector3d.Z;
plot(grains(mineral_name),ipfKey.orientation2color(grains(mineral_name).meanOrientation));
title('Crystal Orientation - Z')

% take a mouse input
disp('Select an individual grain as your seed grain orientation');
[x,y]=ginput(1);

% % hard code the mouse input to run this - swap the commenting around
% % if you want a ginput
% % hard coded position, in um
% x=1379; y=1331;

%add the point to the figure as a black spot
hold on;
scatter(x,y,100,'k','filled');

% find the corresponding grain
grain_sel = grains(x,y);

plot(grain_sel.boundary,'linecolor','g','LineWidth',3,'parent',gca)
plot(grain_sel.boundary,'linecolor','r','LineWidth',1,'parent',gca)
hold off

orientation_threshold=15; %in degrees
% Segment EBSD Cubes based on grain orientation
%1 Define a fibre
% fibre_test = fibre(Miller(1,1,0,ebsd(phasel).CS),yvector);
%2 Extract Orientations using angle function to within 20 degrees of the fibre
% axis (generates a logical array)
test_orientations_2 =
angle(ebsd_2.orientations,grain_sel.meanOrientation,'antipodal')<orientation_threshold*degree
20 degrees is arbitrary
% test_orientations_2 =
angle(ebsd_2.orientations,fibre_test,'antipodal')<20*degree;

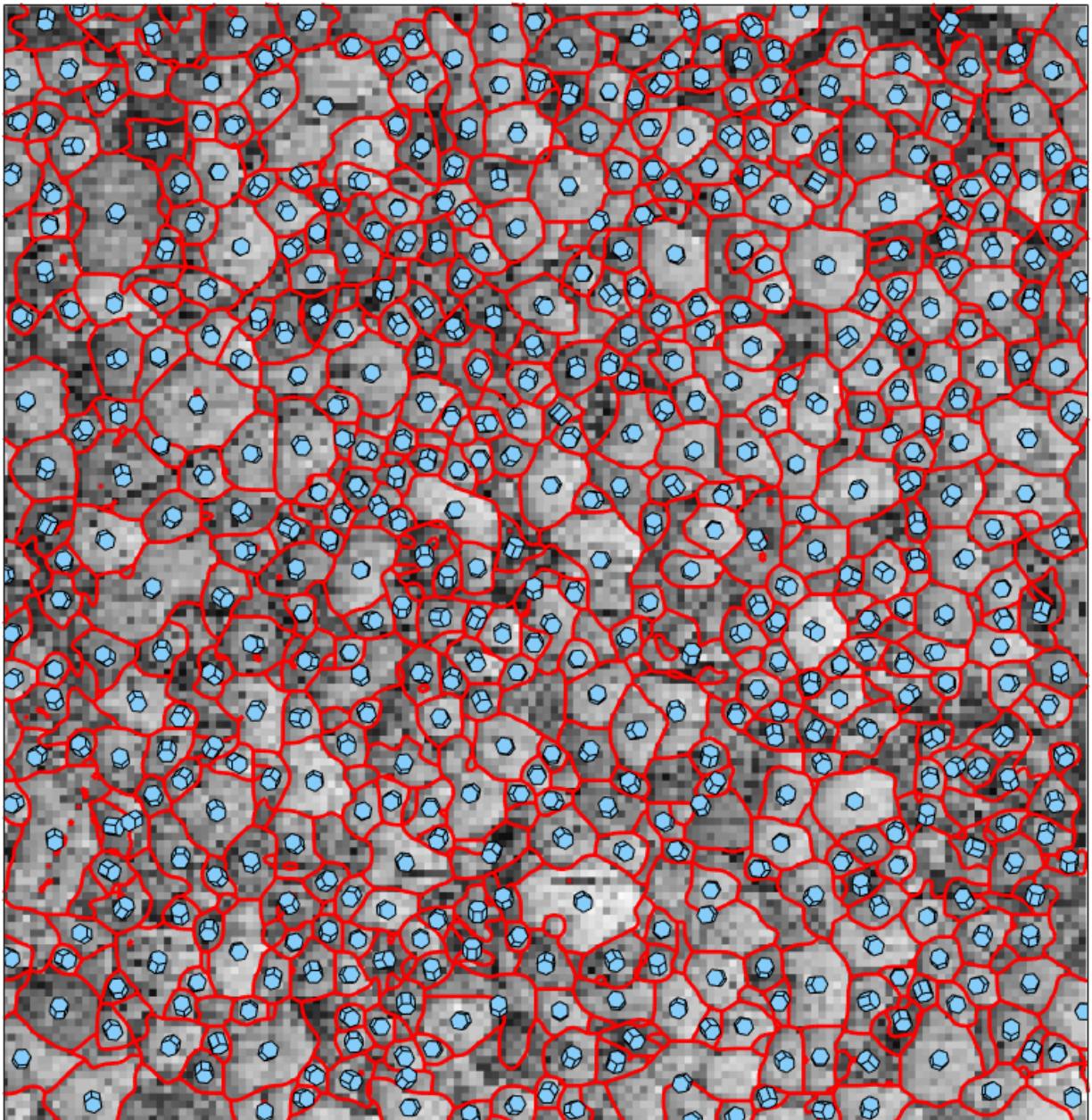
%3 Segment original cube based on Phase (can only do this with one phase)
test_orientations_in = ebsd_2(test_orientations_2 == 1);
test_orientations_out = ebsd_2(test_orientations_2 == 0);

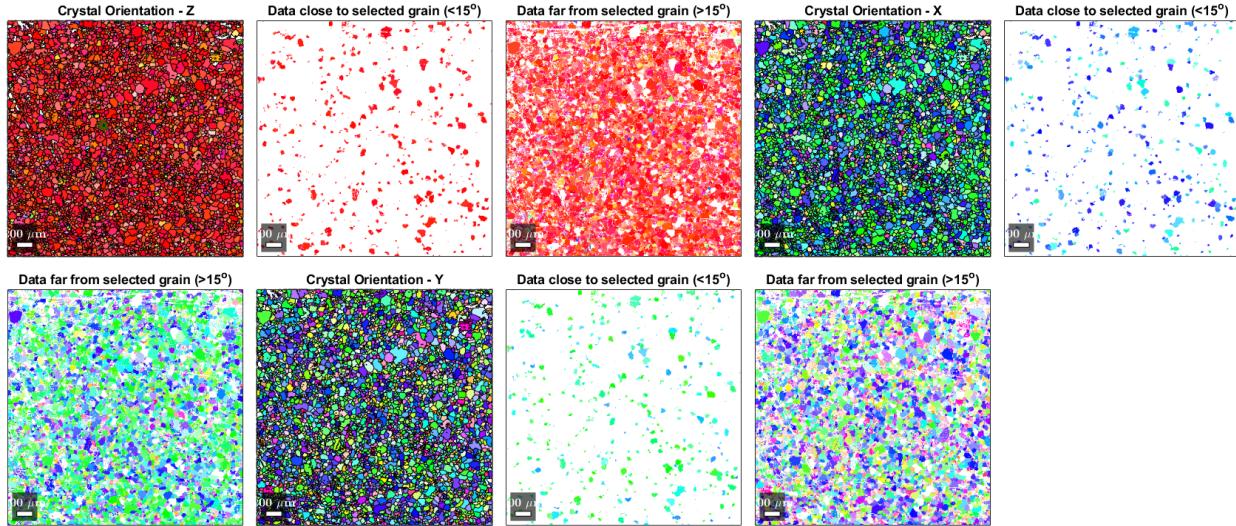
%5 Plot to check
nextAxis;
plot(test_orientations_in,
ipfKey.orientation2color(test_orientations_in.orientations));
title(['Data close to selected grain (<' int2str(orientation_threshold) '^o)' ])
nextAxis;
plot(test_orientations_out,
ipfKey.orientation2color(test_orientations_out.orientations));

```

```
title(['Data far from selected grain (>'  
int2str(orientation_threshold) '^o)''])  
  
ipfKey.inversePoleFigureDirection = vector3d.X;  
nextAxis;  
plot(grains(mineral_name),ipfKey.orientation2color(grains(mineral_name).meanOrientation));  
title('Crystal Orientation - X')  
nextAxis;  
plot(test_orientations_in,  
 ipfKey.orientation2color(test_orientations_in.orientations));  
title(['Data close to selected grain (<'  
 int2str(orientation_threshold) '^o)''])  
nextAxis;  
plot(test_orientations_out,  
 ipfKey.orientation2color(test_orientations_out.orientations));  
title(['Data far from selected grain (>'  
 int2str(orientation_threshold) '^o)''])  
  
ipfKey.inversePoleFigureDirection = vector3d.Y;  
nextAxis;  
plot(grains(mineral_name),ipfKey.orientation2color(grains(mineral_name).meanOrientation));  
title('Crystal Orientation - Y')  
nextAxis;  
plot(test_orientations_in,  
 ipfKey.orientation2color(test_orientations_in.orientations));  
title(['Data close to selected grain (<'  
 int2str(orientation_threshold) '^o)''])  
nextAxis;  
plot(test_orientations_out,  
 ipfKey.orientation2color(test_orientations_out.orientations));  
title(['Data far from selected grain (>'  
 int2str(orientation_threshold) '^o)''])
```

Select an individual grain as your seed grain orientation





plot some unit cells for this

```

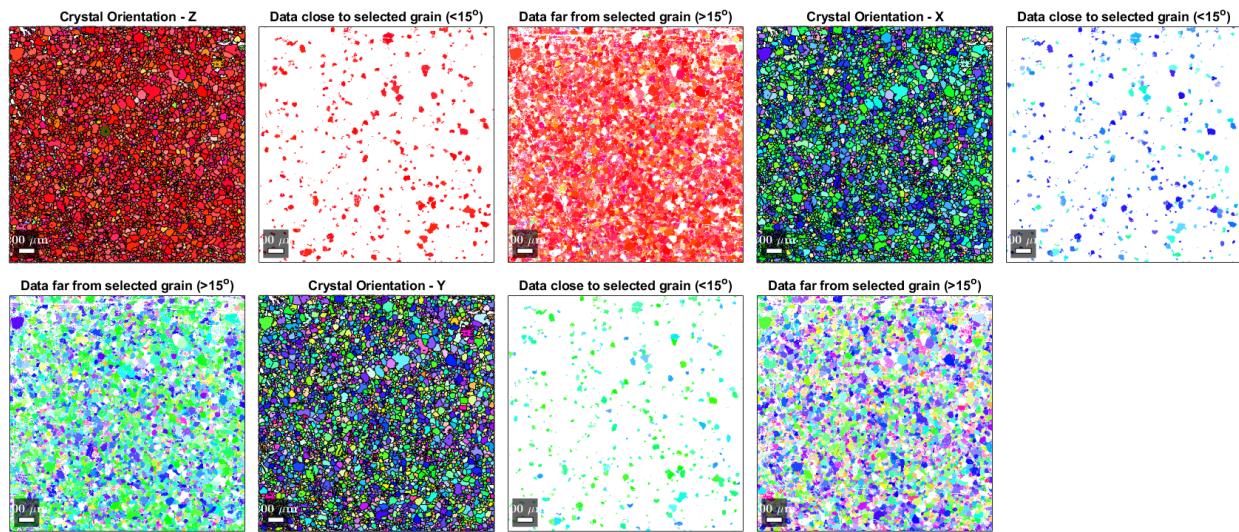
grain_IDs_close=unique(test_orientations_in.grainId);
grains_close=grains(grain_IDs_close);
grains_close=grains_close(grains_close.grainSize>3);
grains_close=grains_close(~grains_close.isBoundary);

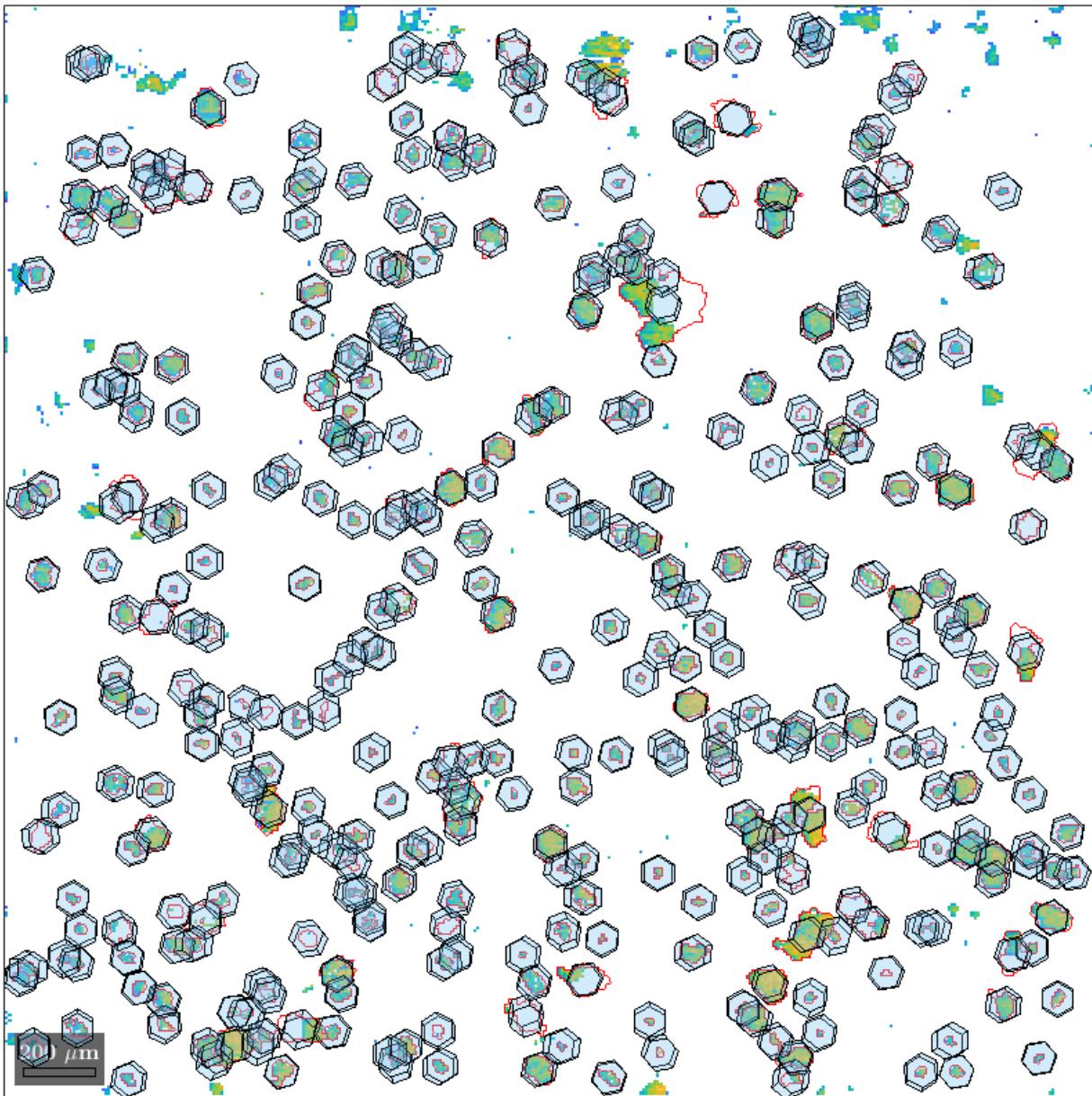
figure;
plot(test_orientations_in, test_orientations_in.prop.Band_Contrast);
% title(['Data far from selected grain (>' int2str(orientation_threshold)
% '^o)'])

hold on
plot(grains_close.boundary, 'linewidth',1,'lineColor','r')

% plot the unit cells
%plot the crystal shape for this point
scaling = 125; % scale the crystal shape to have a nice size
%here we use a hexagon, but you could use a cube for cubic
%crystalShape.cube
cS = crystalShape.hex(grains_close(mineral_name).CS);
grain_centroids=grains_close.centroid;
plot(grain_centroids(:,1),grain_centroids(:,2),50,
grains_close.meanOrientation * cS * scaling, 'FaceAlpha',0.2); %make the
hexagons also transparent so you can see the grains behind

```





copy this m file over, for archival purposes

```
mf_long=mfilename('fullpath');
[f1,f2,f3]=fileparts(mf_long);
mf_start=[mf_long '.m'];
mf_end=fullfile(resultsdir,[f2 '.m']);
try
copyfile(mf_start,mf_end)
catch
warning('m file not saved, likely due to spaces in the file name');
end
```

