MTEX Demo by Dr Ben Britton, Imperial College London

Table of Contents

Contact b.britton@imperial.ac.uk

Zirconium data courtesy of Dr Vivian Tong Exercise working in MTEX 5.2.beta2

Created for the Chemnitz MTEX workshop 2019

Clear variables and tidy up

```
clear
close all
home
```

data and file variables

```
%mtex path - CHANGE
```

```
mtexpath='C:\Users\bbrit\Documents\GitHub\mtex';

% path with h5 file stored - CHANGE
pname = 'C:\Users\bbrit\Documents\GitHub\mtex_demo';
% file to be imported
fname = [pname '\Ax2_1_800N_VT_TBB.h5'];
```

start up MTEX

```
addpath(mtexpath);
startup_mtex
```

```
initialize MTEX 5.2.beta2  .... done!

<strong>MTEX 5.2.beta2</strong> (<a href="matlab:MTEXdoc('mtex')">show
 documentation</a>)
  <a href="matlab:import_wizard('PoleFigure')">Import pole figure
 data</a>
  <a href="matlab:import_wizard('EBSD')">Import EBSD data</a>
  <a href="matlab:import_wizard('ODF')">Import ODF data</a>

  <a href="matlab:uninstall_mtex">Uninstall MTEX</a>
```

Establish plotting convention - Bruker & Imperial Checked

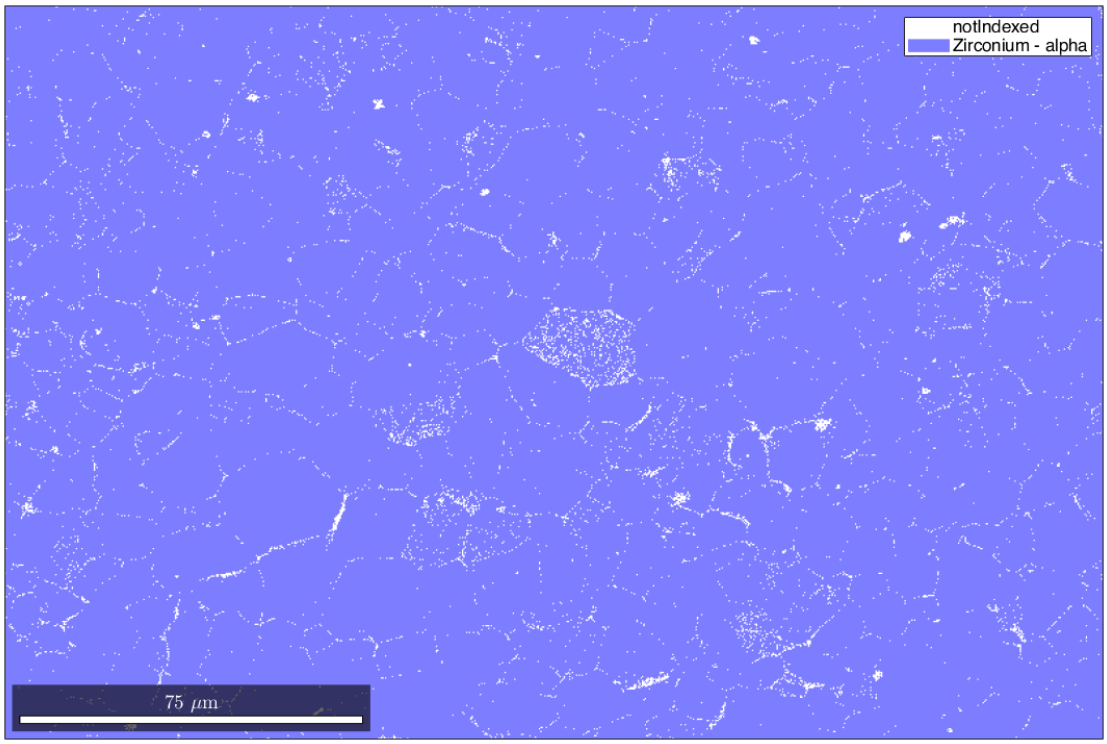
```
setMTEXpref('xAxisDirection','west');
setMTEXpref('zAxisDirection','outOfPlane');
```

Load the data

```
% create an EBSD variable containing the data
[ebsd,header]=loadEBSD_h5v2(fname);

%convert into an XY grid to make life easier
ebsd=ebsd.gridify;
```

Start plotting

```
figure; %create a new figure window
plot(ebsd); %plot the EBSD data
```

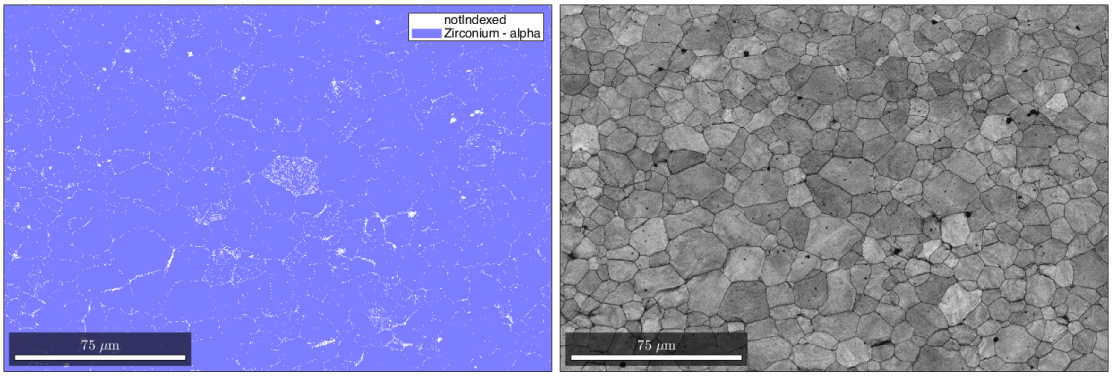


extract the mineral name of phase == 1

```
%find the points in the map which correspond to phase 1 (Zr for this
 map)
phasepts=find(ebsd.phase == 1);
%extract the mineral for one point
phase=ebsd(phasepts(1)).mineral;

clear phasepts %clear this temporary variable - makes the Workspace
 tidier
```

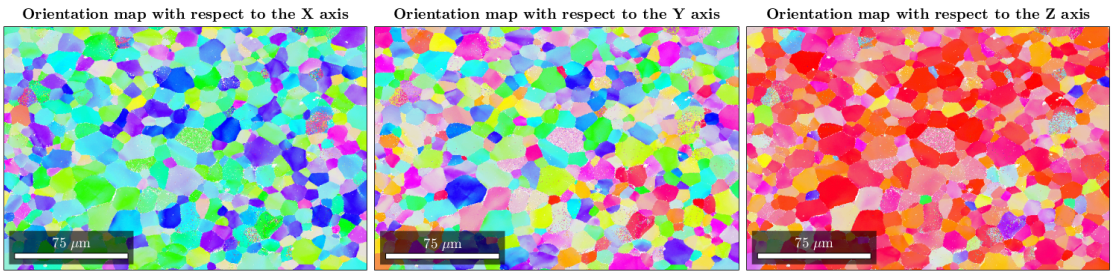
plot the quality map

```
% enables us to see how this data looks
nextAxis %create a new axis on the existing figure and put along side
plot(ebsd,ebsd.prop.RadonQuality);
colormap('gray')
```



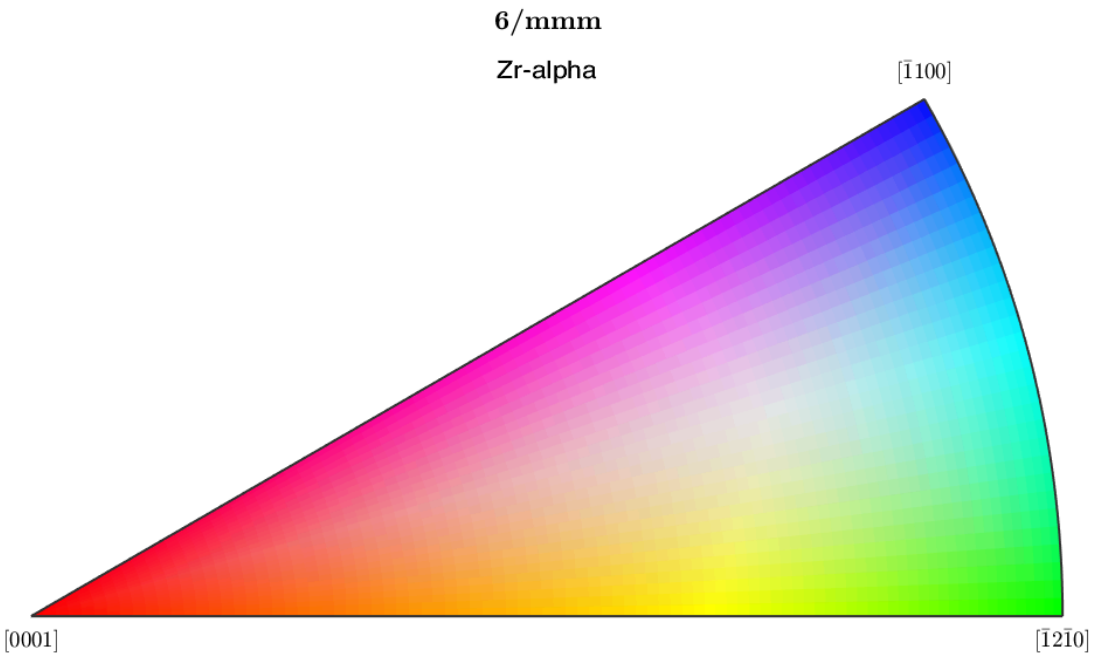
Plot the EBSD maps based upon IPF colouring

```
%create the colourkey
oM1=ipfHSVKey(ebsd(phase));

%plot the figure
figure;
oM1.inversePoleFigureDirection=xvector; %IPFx wrt X
plot(ebsd(phase),oM1.orientation2color(ebsd(phase).orientations));
mtexTitle('Orientation map with respect to the X axis')

nextAxis
oM1.inversePoleFigureDirection=yvector; %IPFx wrt Y
plot(ebsd(phase),oM1.orientation2color(ebsd(phase).orientations));
mtexTitle('Orientation map with respect to the Y axis')

nextAxis
oM1.inversePoleFigureDirection=zvector; %IPFx wrt Z
plot(ebsd(phase),oM1.orientation2color(ebsd(phase).orientations));
mtexTitle('Orientation map with respect to the Z axis')
```



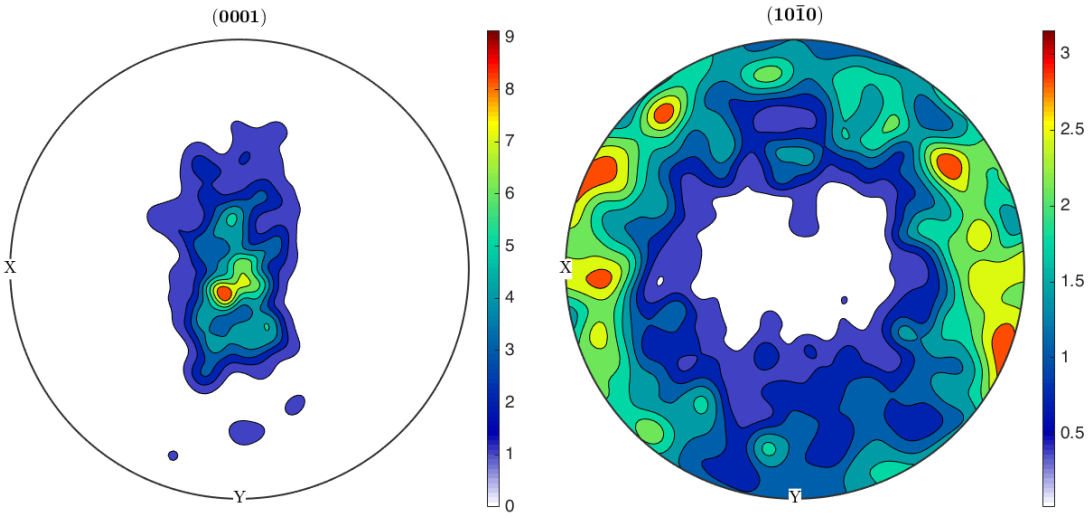
Plot the colour key

```
figure('Color',[1 1 1]);
plot(oM1); title('Zr-alpha');
```



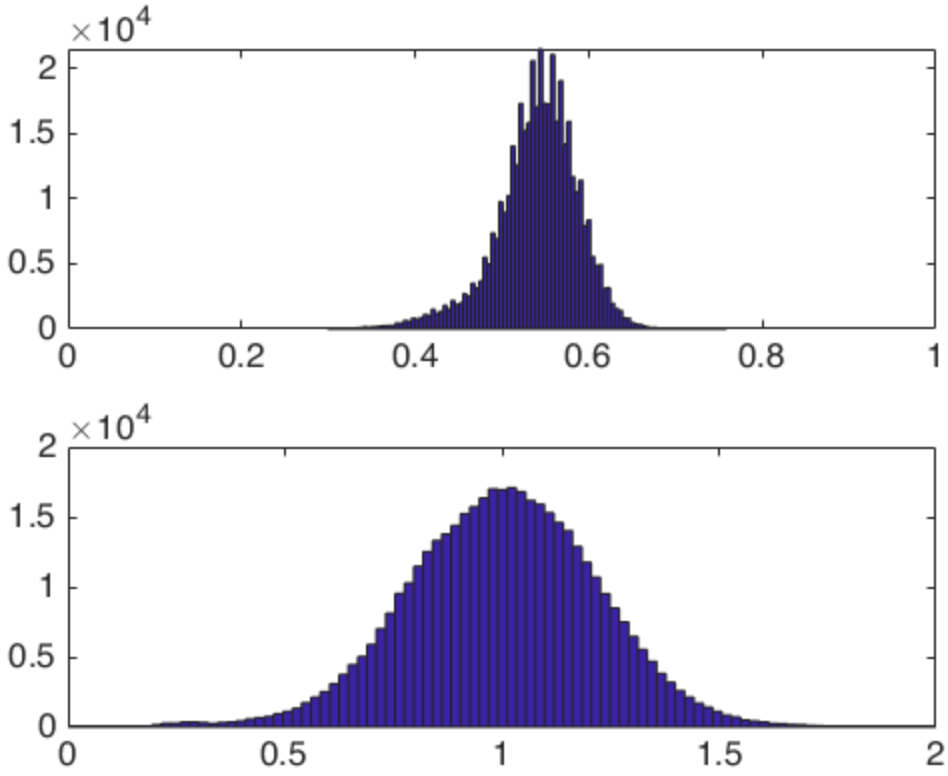
Plot the texture using the ODF

```
odf_width=5; %in degrees

odf = calcODF(ebsd(phase).orientations,'halfwidth',odf_width*degree);

h = Miller({0,0,1},{1,0,0},odf.CS); %plot the (001) and (100), i.e.
 basal and prism, plane ODFs

figure;
plotPDF(odf,h,'upper','projection','eangle','contourf');
%Note you can plot ODFs with different projections & fix the
 colourscales
mtexColorbar;
```



Threshold some data based upon quality

```
% first inspect the distributions
figure;
subplot(2,1,1); %[yboxes,xboxes,boxnum]
hist(ebsd.prop.RadonQuality(:),100); %The Hough based quality;
%needs the (:) on the end to create a column grid
xlim([0 1]);

subplot(2,1,2);
hist(ebsd.prop.MAD(:),100); %The Hough mean angular deviation in
 degrees, for Bruker
xlim([0 2]);
```



Threshold these values

```
Thresh_RadonQ=0.4; %RadonQuality
Thresh_MAD=2; %radon limit, upper

ebsd_good=ebsd(phase); %extract only the Zr-alpha
ebsd_good=ebsd_good(ebsd_good.prop.RadonQuality > Thresh_RadonQ);
ebsd_good=ebsd_good(ebsd_good.prop.MAD < Thresh_MAD);

% re-grid
ebsd_good=ebsd_good.gridify;
```

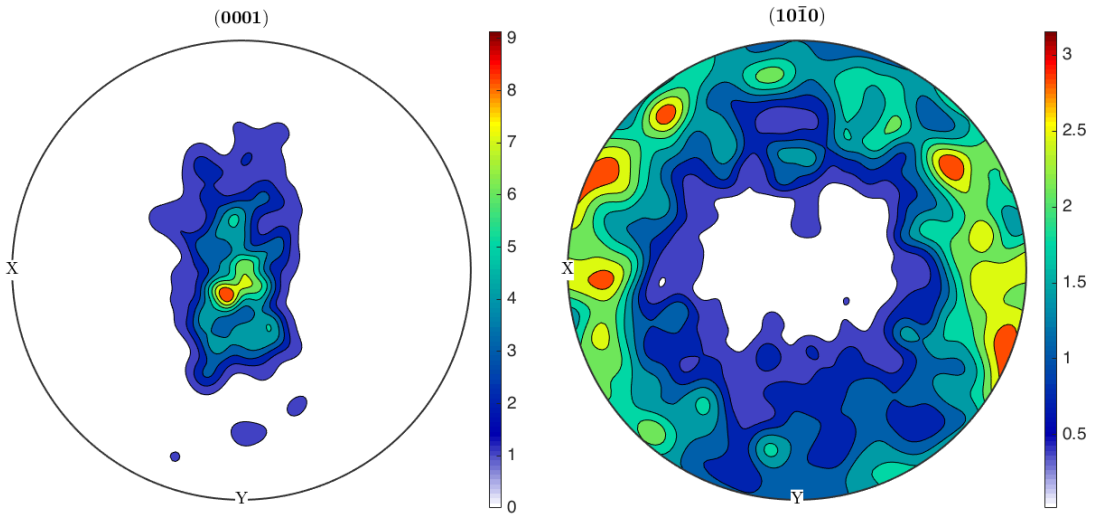
Now calculate the grains

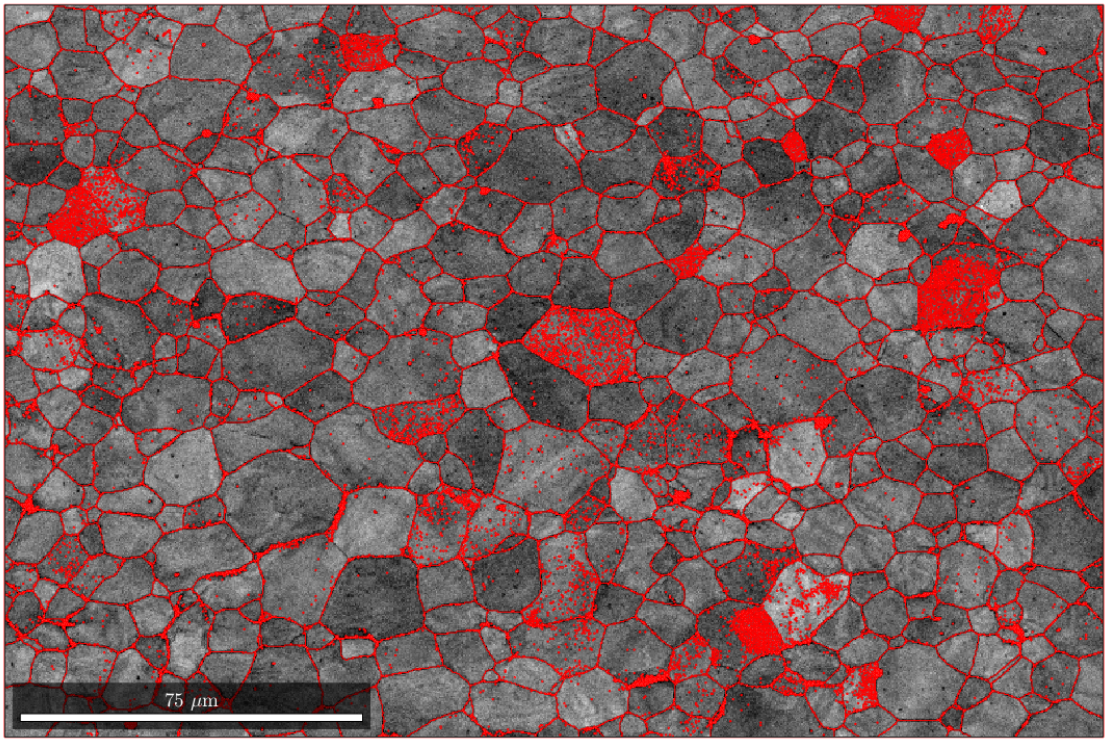
```
gbThreshold = 5*degree;
[grains,ebsd_good.grainId]=calcGrains(ebsd_good('indexed'),'angle',gbThreshold);
```

plot the grain boundary map over the quality map to check that this looks reasonable

```
figure;
plot(ebsd_good,ebsd_good.prop.RadonQuality); colormap('gray');
hold on;
```

```
%add on the grain boundaries
plot(grains.boundary,'linewidth',0.5,'lineColor','r');
```



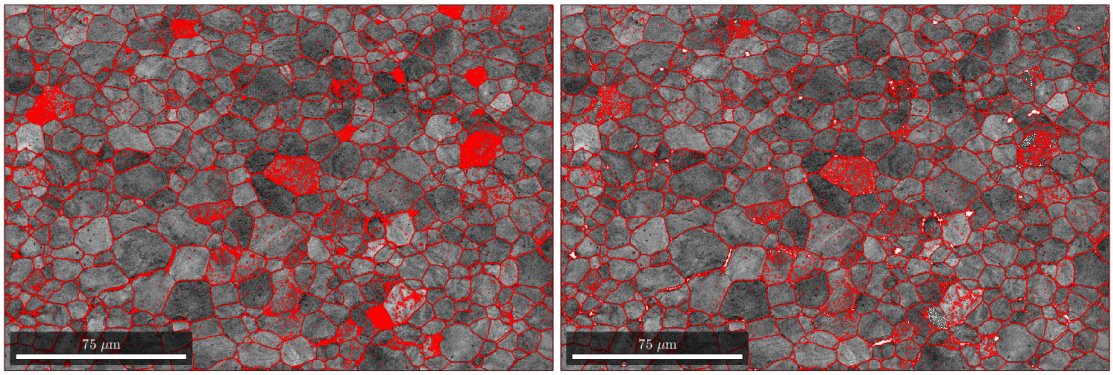


Remove the small grains from the list

```
num_pixel=10; %threshold number of pixels

%remove small pixel grains
grains_big=grains(grains.area > num_pixel*header.XSTEP*header.YSTEP);
ebsd_good_big=ebsd_good(ebsd_good(grains_big));
ebsd_good_big=ebsd_good_big.gridify;
```

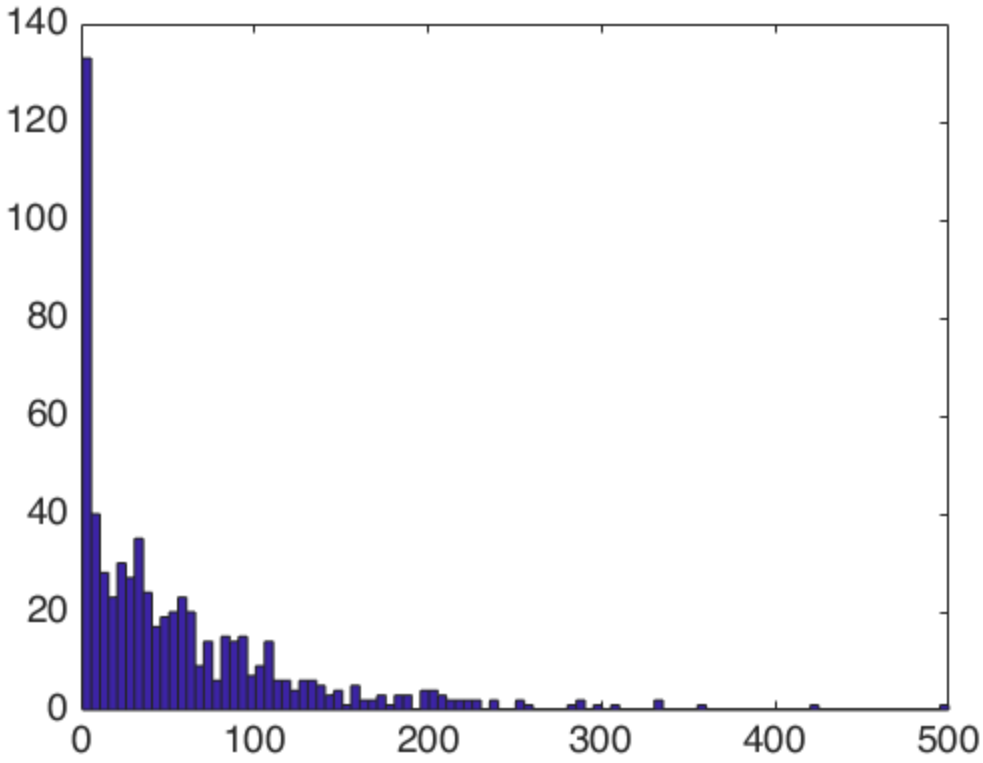
Plot on the previous map

```
nextAxis;
plot(ebsd_good_big,ebsd_good_big.prop.RadonQuality); colormap('gray');
hold on;
%add on the grain boundaries
plot(grains_big.boundary,'linewidth',0.5,'lineColor','r');
```



Histogram the grain size

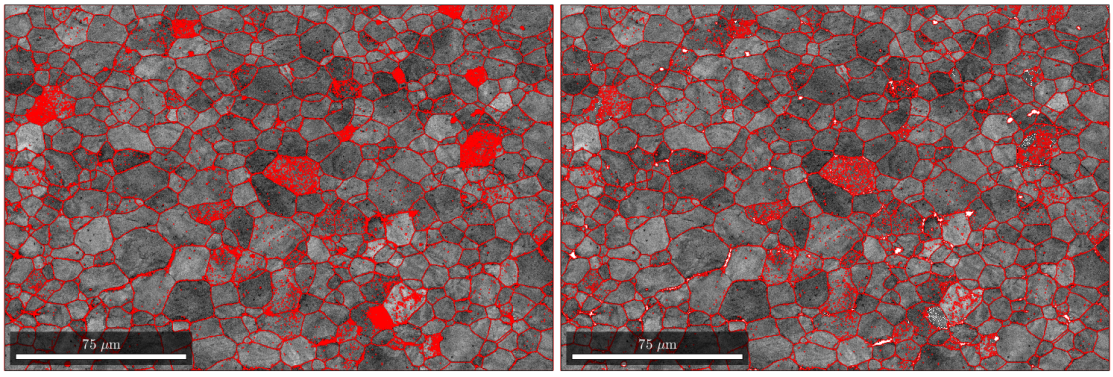
```
%histogram on grain size
figure;
hist(grains_big.area,100); %100 bins for the histogram
```

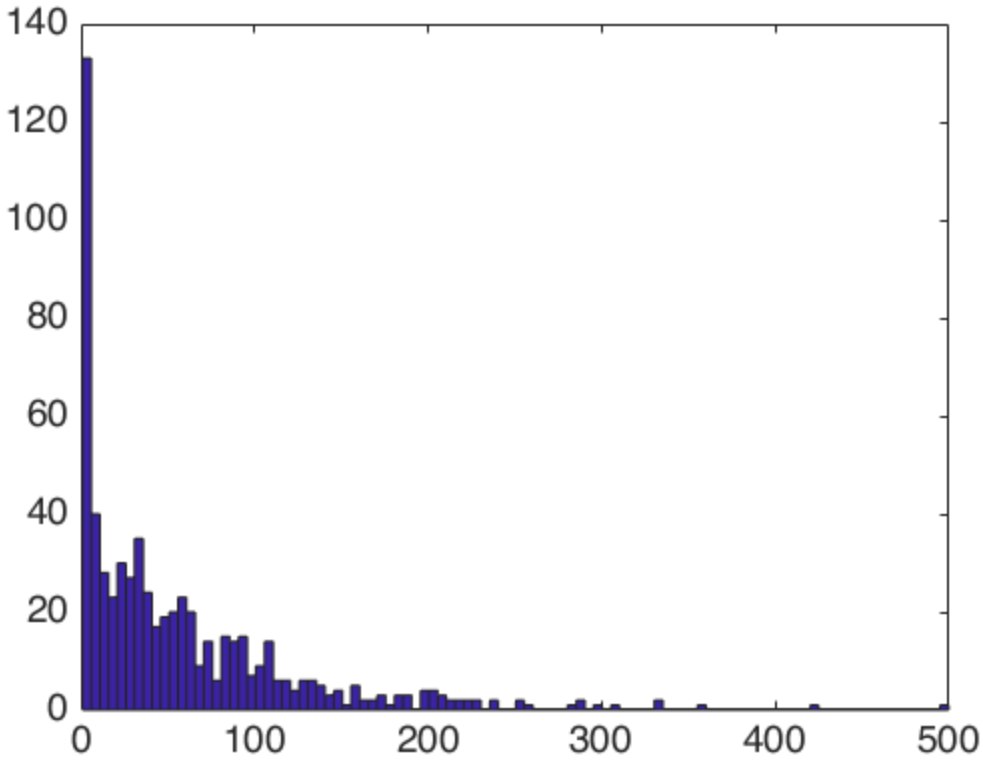


Smooth the data - USE WITH CARE

This is pretty and easier to work with Be very careful about the smoothing function and the structure inherited

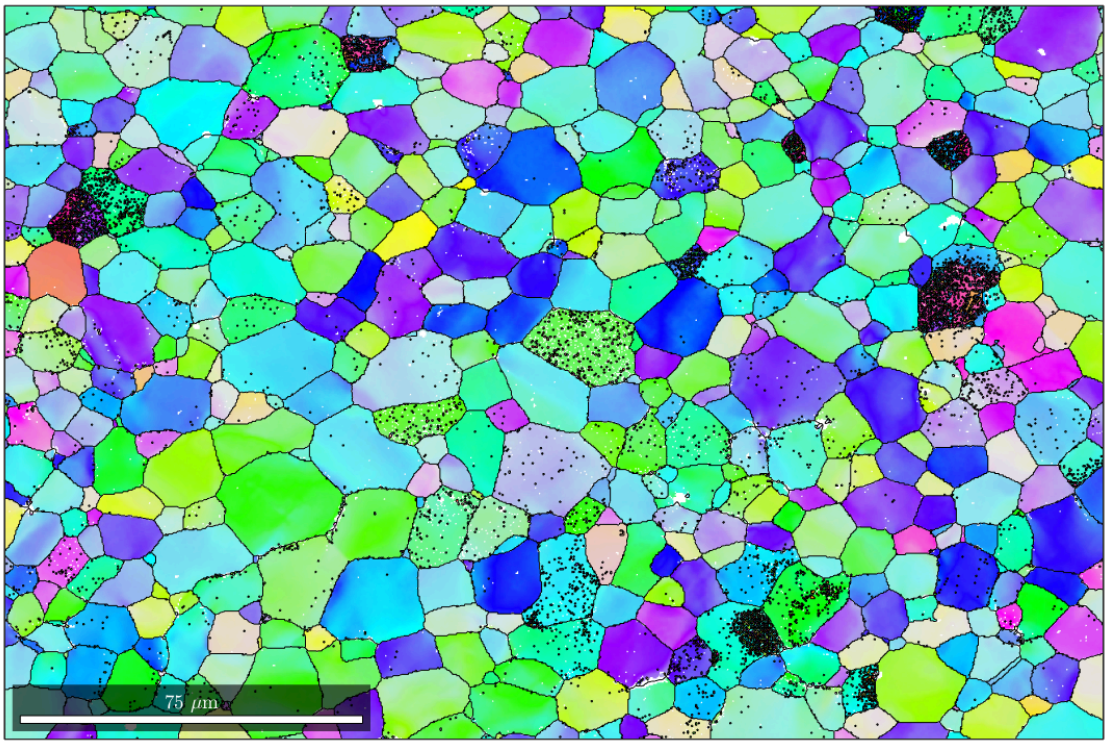
```
F = meanFilter; %pick the spline points
ebsd_smoothed = smooth(ebsd_good_big,F,'fill',grains_big); %this is
 still on a grid - but you can always check

%recalc the grains - this prooves useful for later
[grains_smooth,ebsd_smoothed.grainId]=calcGrains(ebsd_smoothed('indexed'),'angle',
```





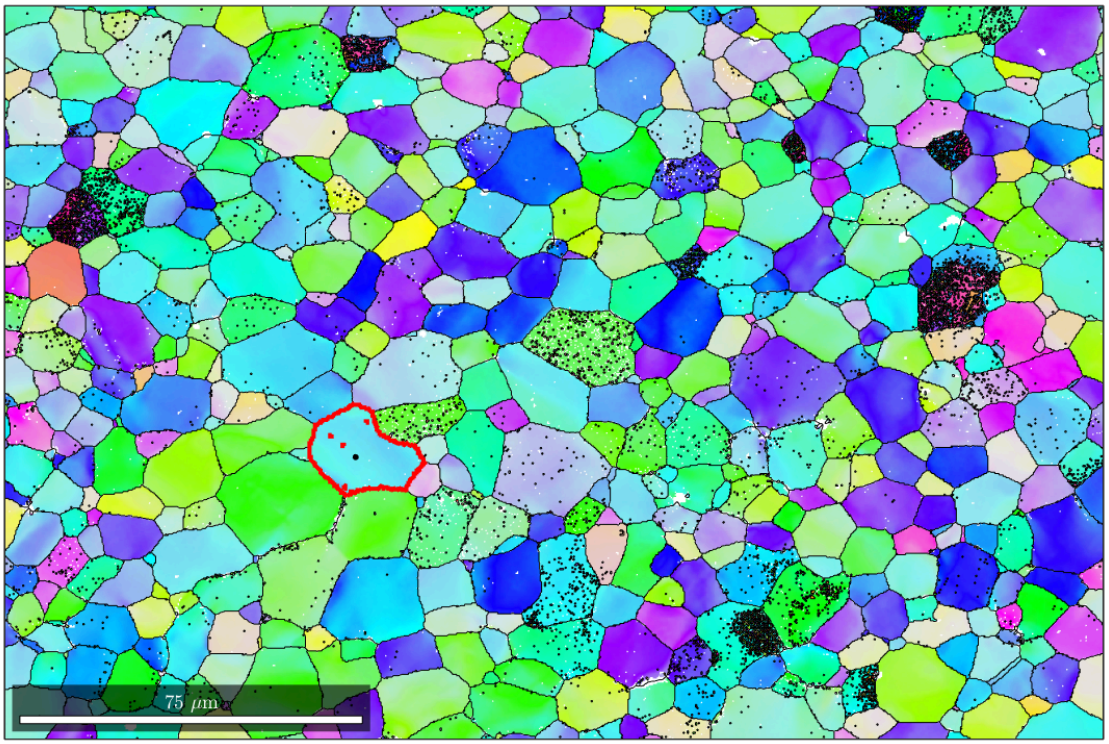
Plot the updated IPF map

```
oM1=ipfHSVKey(ebsd_smoothed(phase));
oM1.inversePoleFigureDirection=xvector; %IPFx

figure;
plot(ebsd_smoothed(phase),oM1.orientation2color(ebsd_smoothed(phase).orientations)
hold on;
plot(grains_smooth.boundary,'linewidth',0.5,'lineColor','k');
```



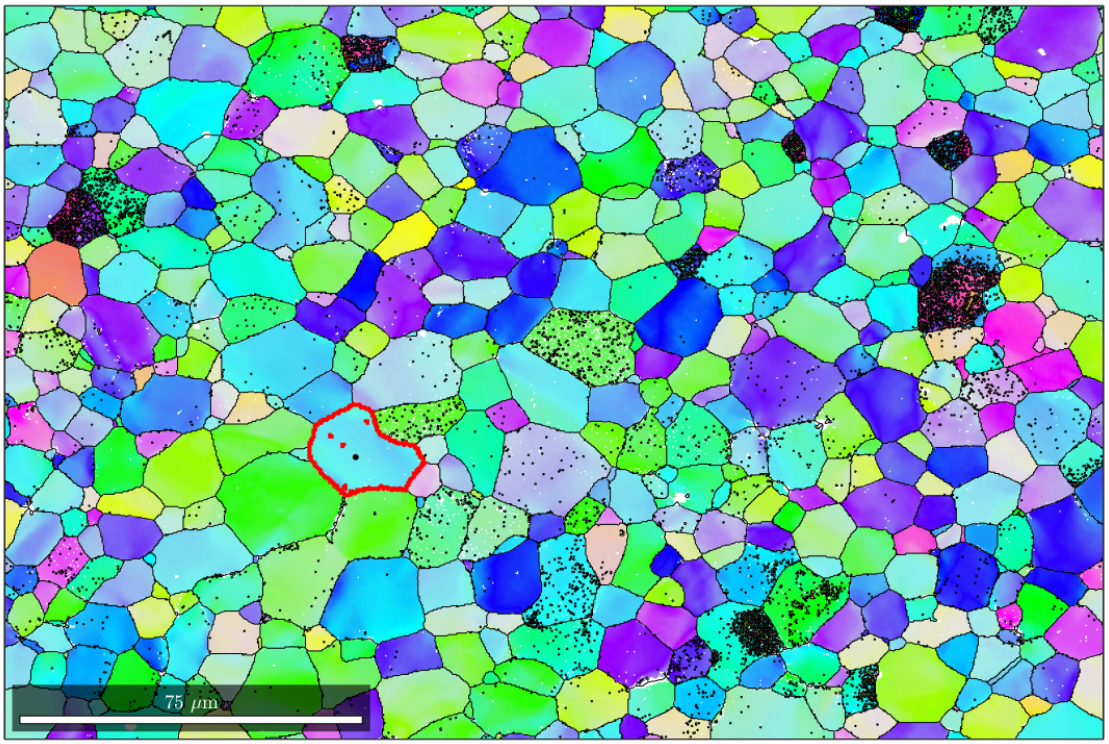
Now we can extract one grain and plot it as an extract

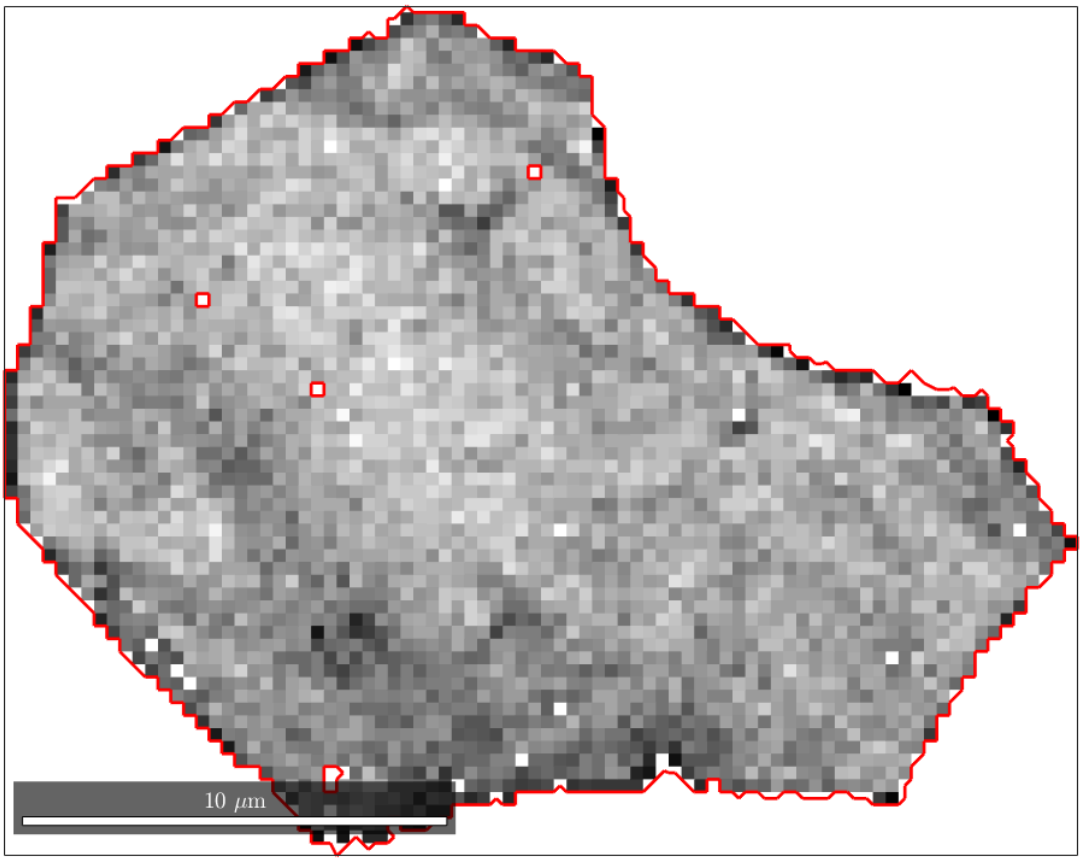
```
[x,y]=ginput(1); %use a mouse cursor to pick a grain
hold on;
scatter(x,y,20,'k','filled');

% find the corresponding grain
grain_sel = grains_smooth(x,y);
plot(grain_sel.boundary,'linecolor','r','LineWidth',3);
hold off
```



Now plot this grain as a single image - useful for showing off this grain

```
figure;

%subset the plot to just have the selected grain
plot(ebsd_smoothed(grain_sel),ebsd_smoothed(grain_sel).prop.RadonQuality);
colormap('gray');
hold on
plot(grain_sel.boundary,'LineWidth',2,'linecolor','r');
```

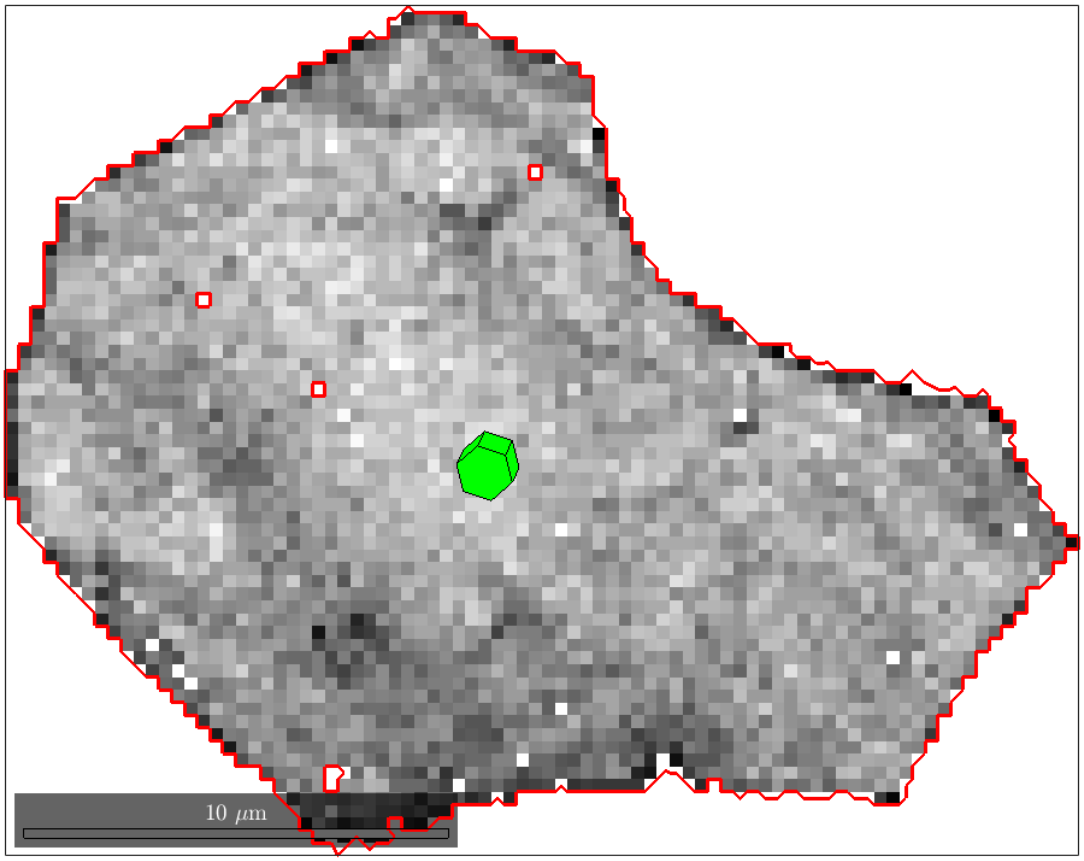

75 μm


10 μm

Add a unit cell

```
%generate the unit cell shape (this is HCP)
cS = crystalShape.hex(ebsd_smoothed(phase).CS);
%plot the crystal - 0.1 = fraction of the grain shape
plot(grain_sel,0.1*cS,'FaceColor','g')
```

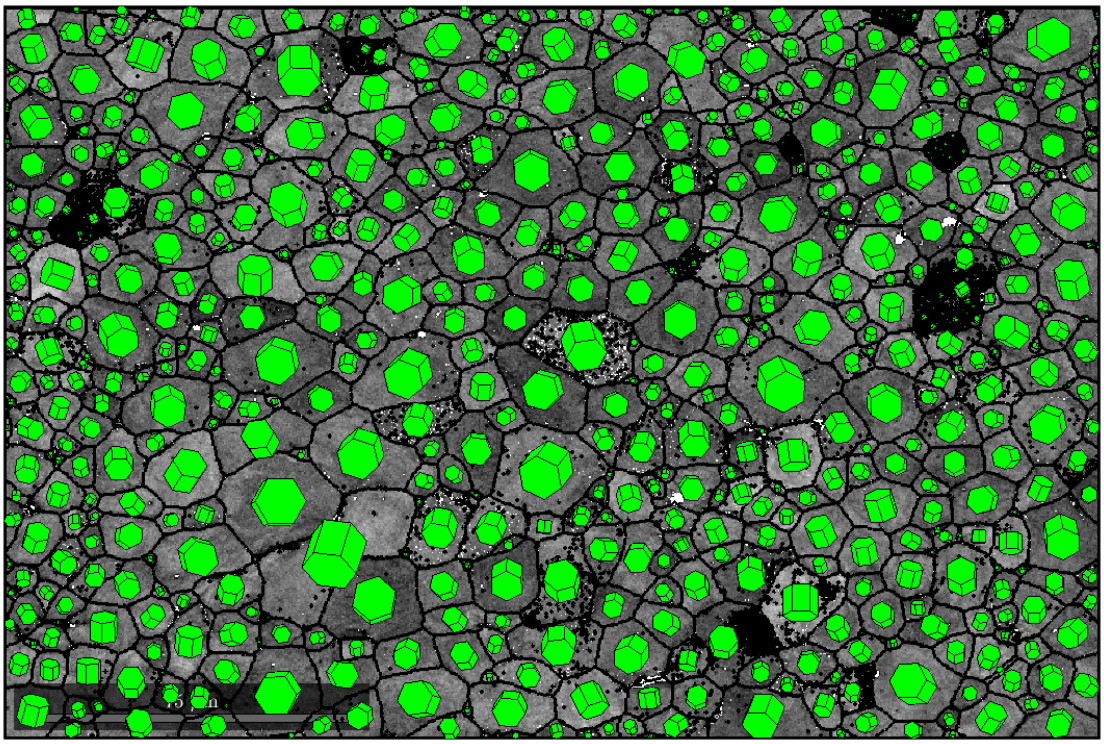
```
Warning: Symmetry missmatch!
Warning: Symmetry missmatch!
Warning: Symmetry missmatch!
Warning: Symmetry missmatch!
Warning: Symmetry missmatch!
Warning: Symmetry missmatch!
Warning: Symmetry missmatch!
Warning: Symmetry missmatch!
Warning: Symmetry missmatch!
Warning: Symmetry missmatch!
Warning: Symmetry missmatch!
```



Can also plot unit cells for the entire map

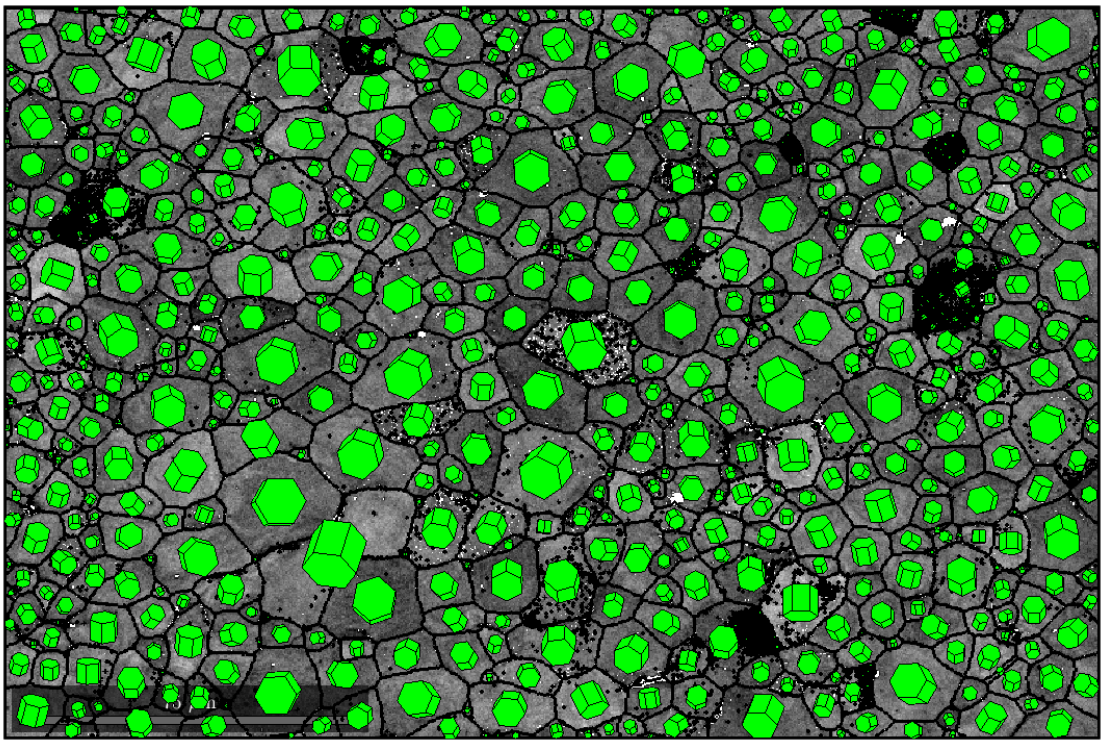
```
figure;
plot(ebsd_smoothed(phase),ebsd_smoothed(phase).prop.RadonQuality);
```

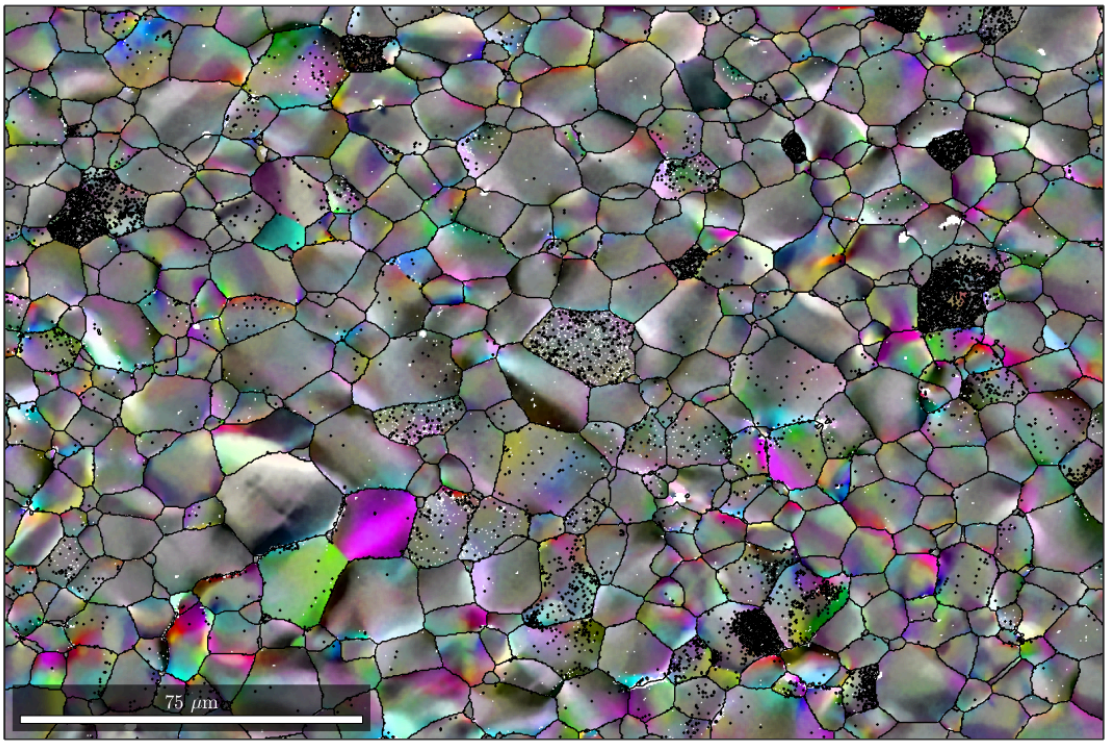
```
colormap('gray');
hold on
plot(grains_smooth.boundary,'LineWidth',2,'linecolor','k');
plot(grains_smooth,0.7*cS,'FaceColor','g')
```

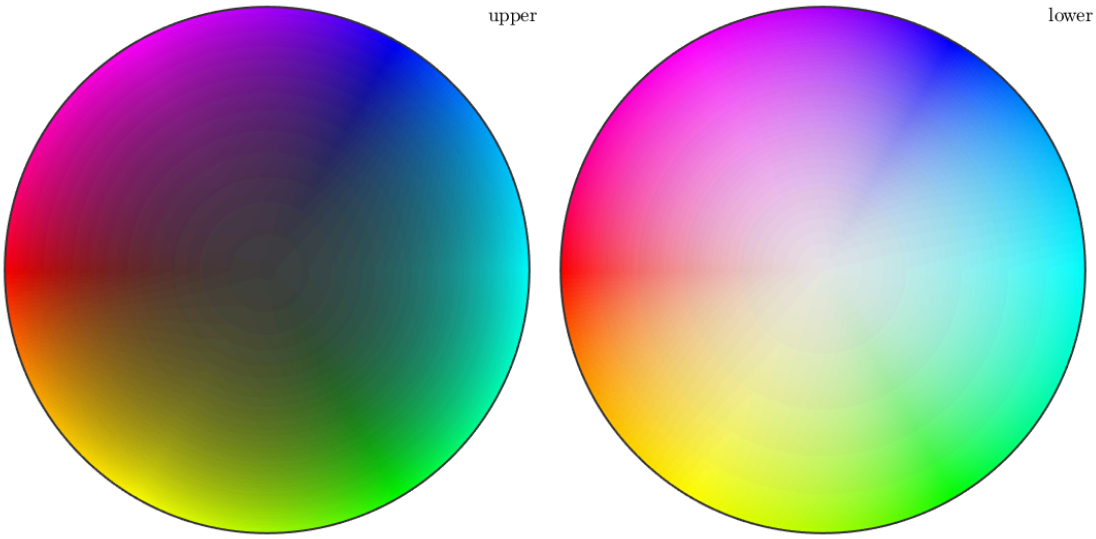


Plot the orientation from the mean - sample coordinates

```
% plot mis2mean for all phases
ipfKey = axisAngleColorKey(ebsd_smoothed(phase));
ipfKey.maxAngle = 5*degree;

%choose the orientation reference for each grain
ipfKey.oriRef =
 grains_smooth.meanOrientation(ebsd_smoothed(phase).grainId);

%plot the map
figure;
plot(ebsd_smoothed(phase),ipfKey.orientation2color(ebsd_smoothed(phase).orientatio
hold on
% plot boundary
plot(grains_smooth.boundary,'linewidth',1)
hold off

%plot the colourkey
figure;
plot(ipfKey);
```




75 μm

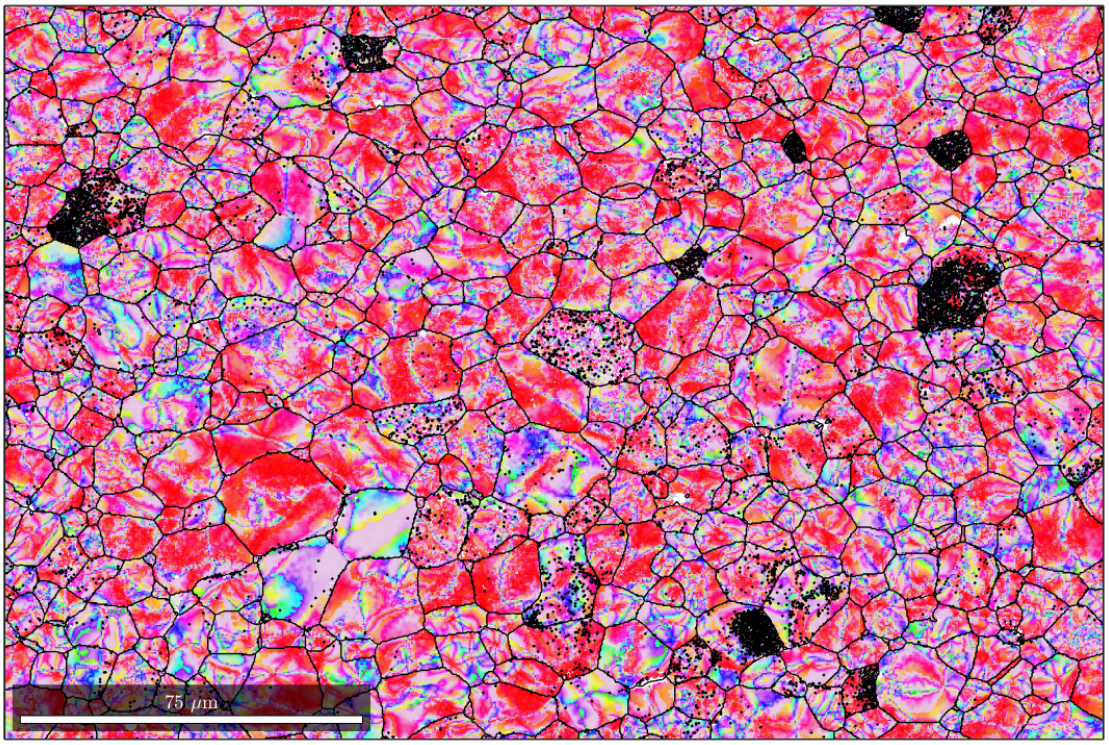


Calculate the misorientation axis & plot in the crystal frame

```
%calculate the misorientation in the specimen frame
axis_specimen=axis(grains_smooth(ebsd_smoothed(phase).grainId).meanOrientation,ebs

%calculate the misorientation in the crystal frame (i.e. rotate each
%according to the grain mean orientation
axis_crystal=axis(inv(grains_smooth(ebsd_smoothed(phase).grainId).meanOrientation)
angle_crystal=angle(inv(grains_smooth(ebsd_smoothed(phase).grainId).meanOrientatio

%create the IPF colour key
HCP_IPFkey=HSVDirectionKey(cS.CS);

%create the colours
RGB=HCP_IPFkey.direction2color(axis_crystal);
```

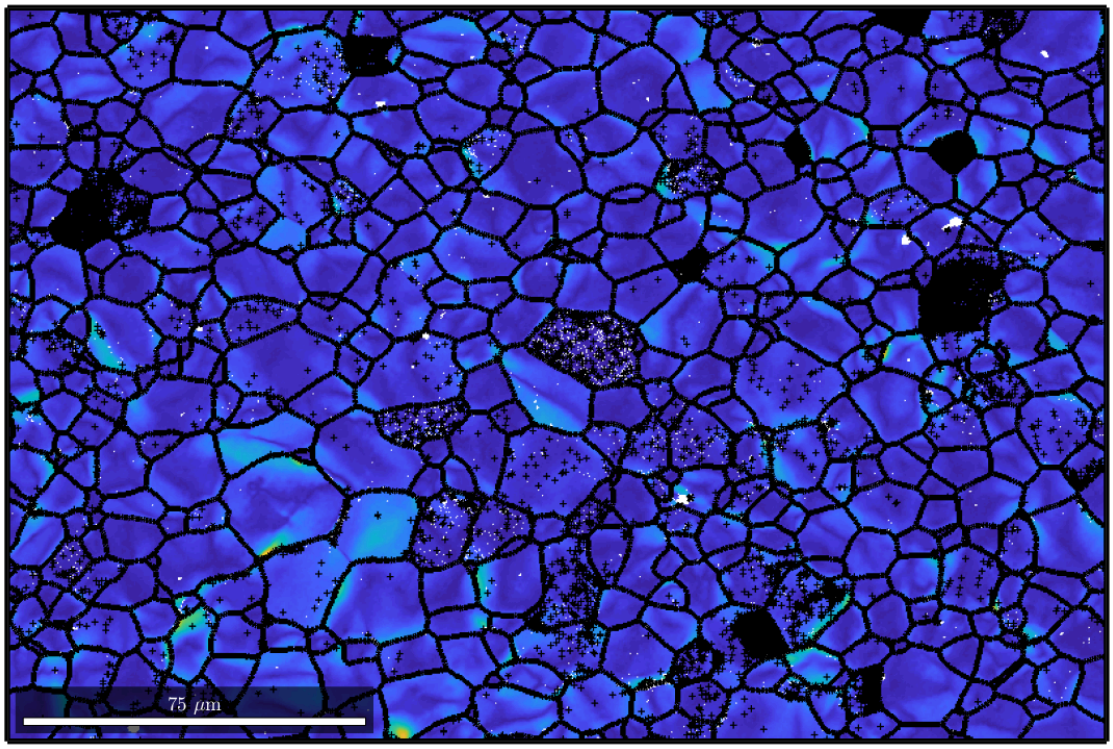
Plot the axis for all the EBSD data

```
figure;
plot(ebsd_smoothed(phase),RGB);
hold on;
% plot boundary
plot(grains_smooth.boundary,'linewidth',1)
hold off
```



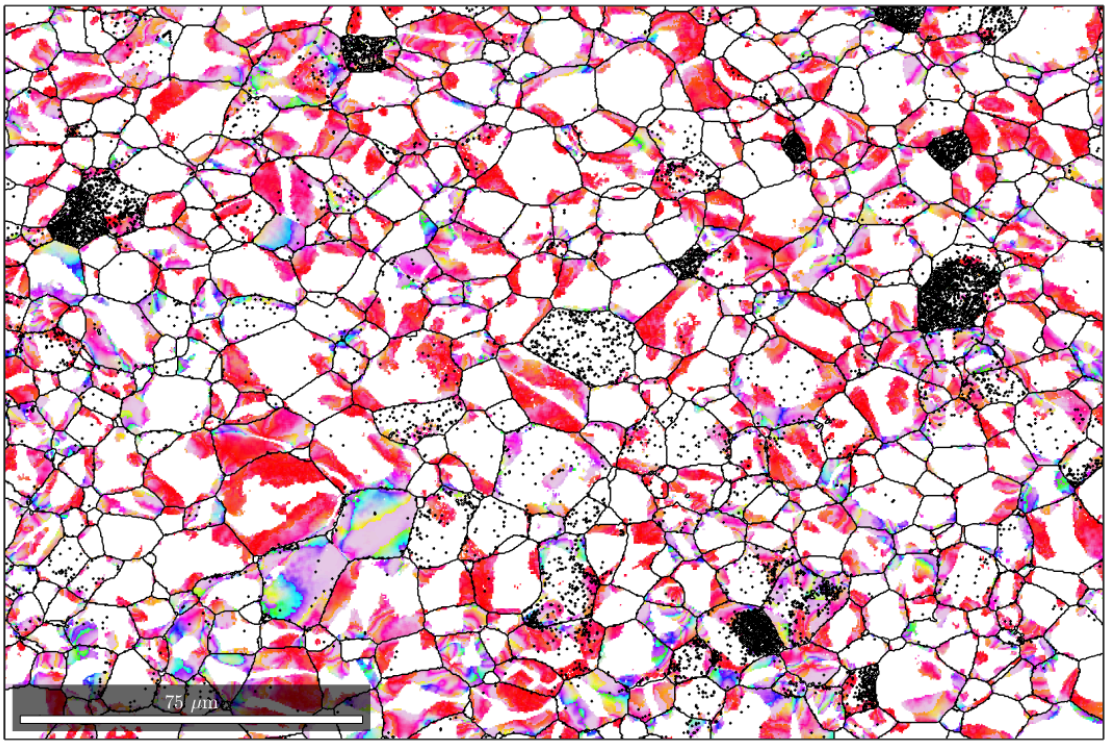
Plot the magnitude of the angle

```
figure;
plot(ebsd_smoothed(phase),angle_crystal);
hold on;
% plot boundary
plot(grains_smooth.boundary,'linewidth',4)
hold off
```



Reduce to plot axes for points with an angle above a threshold

```
ebsd_good=ebsd_smoothed(phase);
ebsd_good=ebsd_good(angle_crystal>1.5*degree);
RBG_reduced=RGB(angle_crystal>1.5*degree,:);

figure;
plot(ebsd_good(phase),RBG_reduced);
hold on;
% plot boundary
plot(grains_smooth.boundary,'linewidth',1)
hold off

%end of script
```



Published with MATLAB® R2018b