Table of Contents


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

Import Script for EBSD Data

This script was automatically created by the import wizard. You should run the whoole script or parts of it in order to import your data. There is no problem in making any changes to this script.

```
%location of MTEX 5.10.2
%if MTEX loaded this doesn't matter
%if MTEX is not loaded, then this will be used to start MTEX up
mtex_location='C:\Users\ruthb\OneDrive\Documents\MTEX\mtex-5.11.2';
% mtex_location='/MATLAB Drive/mtex-5.10.2';

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

Specify Crystal and Specimen Symmetries

```
% crystal symmetry
CS = {...
  'notIndexed',...
  crystalSymmetry('SpaceId',194, [3.2 3.2 5.2], 'X||a', 'Y||b*', 'Z||
c*', 'mineral', 'Magnesium', 'color', [0.53 0.81 0.98])};

%use this mineral for plotting
mineral_name=CS{2}.mineral;

% plotting convention
```

```
setMTEXpref('xAxisDirection','east');
setMTEXpref('zAxisDirection','inToPlane');
```

Specify File Names

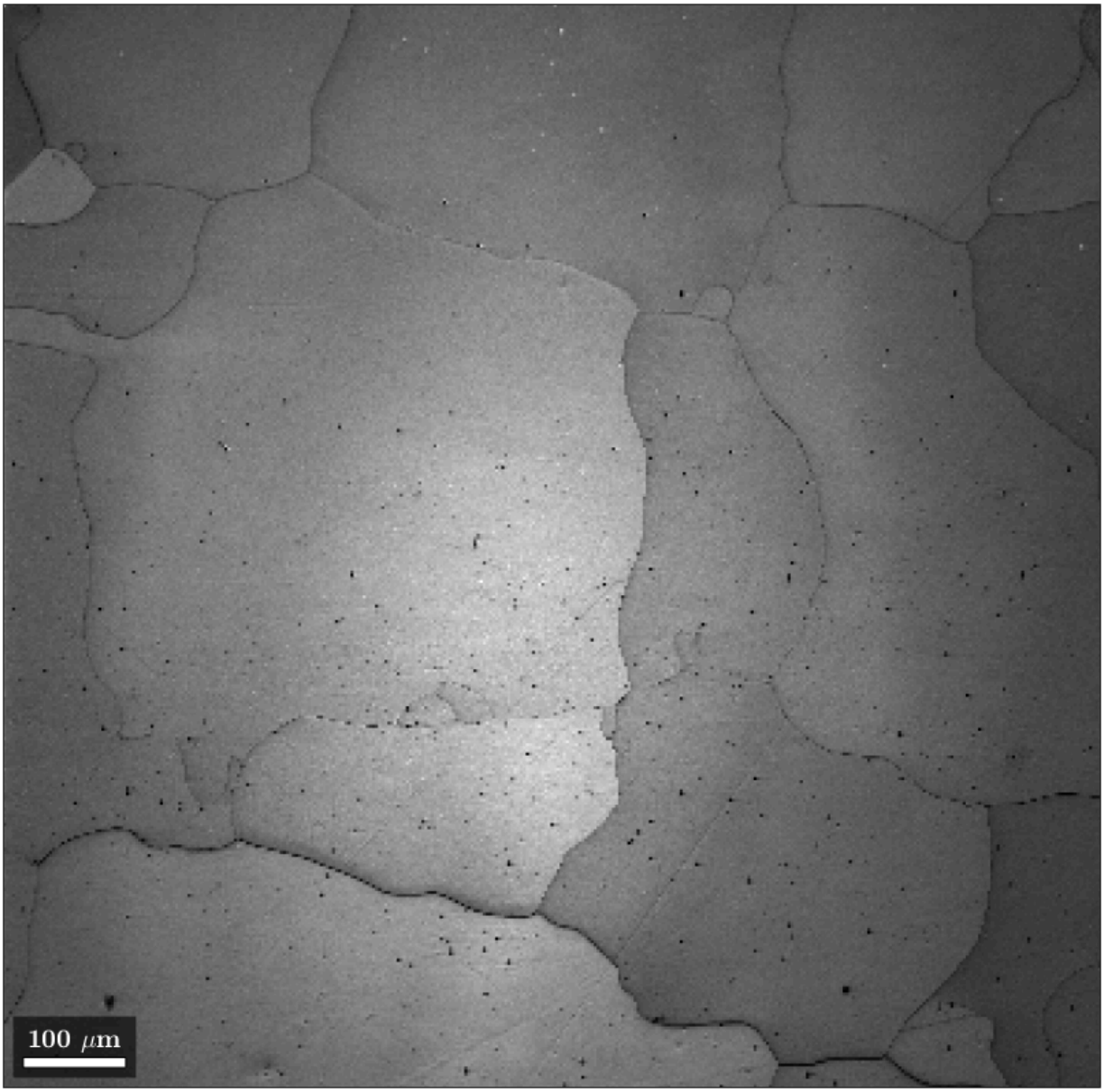
```
% path to files
pname = 'C:\Users\ruthb\DocumentsOnLaptop\GitHub\MTEX_Workshop\Data';

% which files to be imported
fname = [pname '\scan1.ang'];
```

Import the Data

```
% create an EBSD variable containing the data
% ebsd = EBSD.load(fname,CS,'interface','ang',...
%   'convertEuler2SpatialReferenceFrame','setting 1');

ebsd = EBSD.load(fname,CS,'interface','ang',...
  'convertEuler2SpatialReferenceFrame','setting 2');
```

Do some plots

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

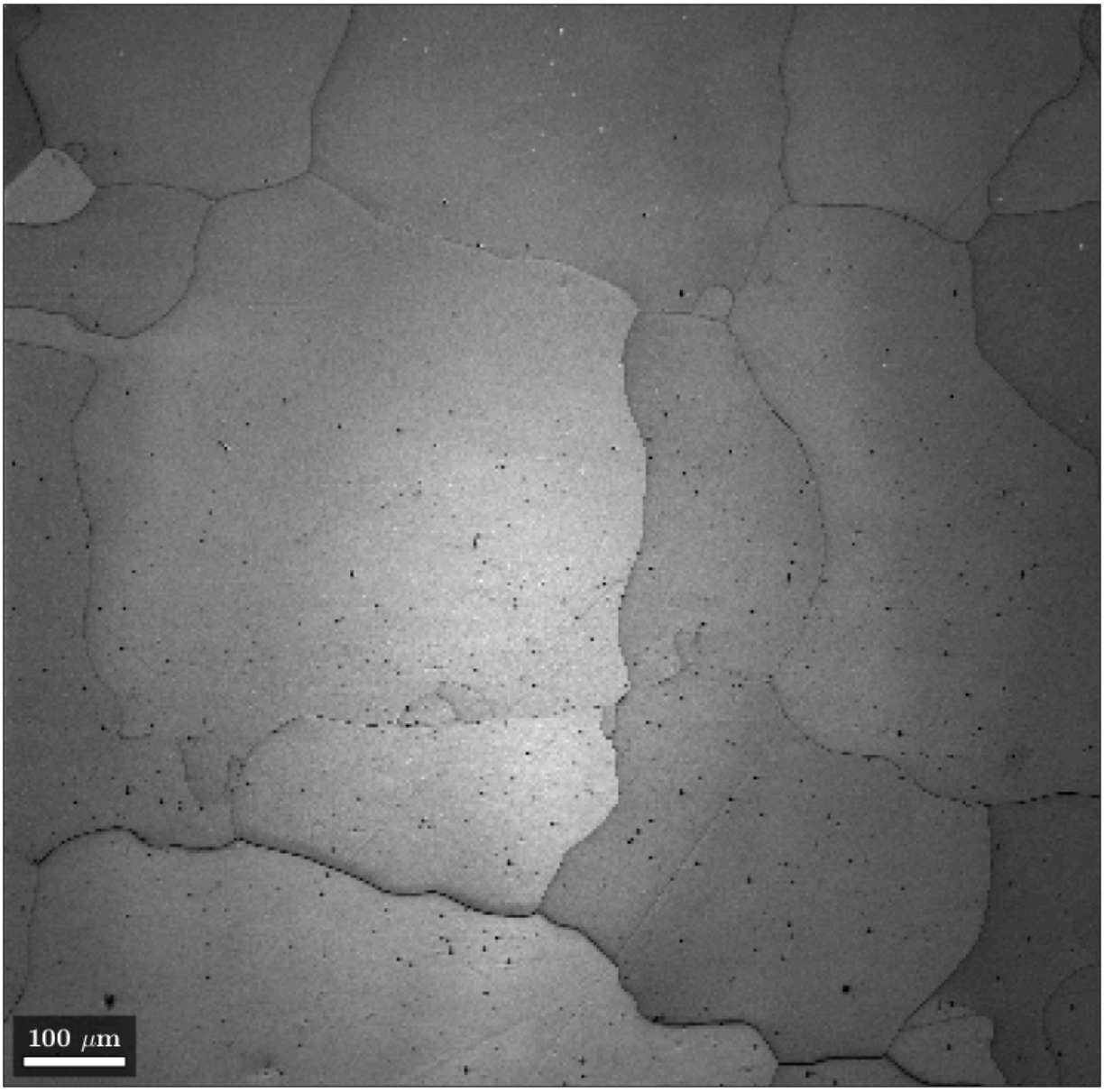


IPF-x, y and z

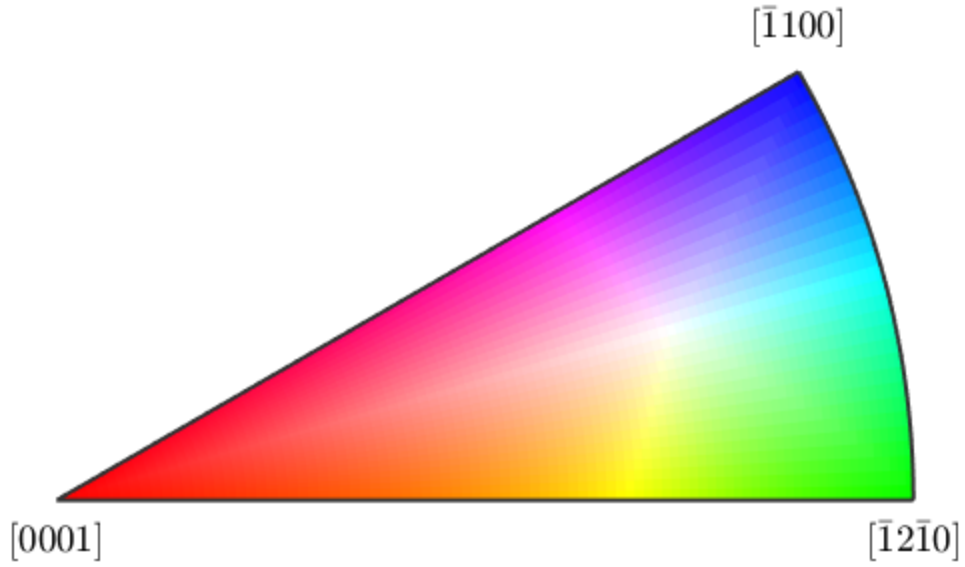
compute the colors ipfKey = ipfHSVKey(ebsd(mineral_name));

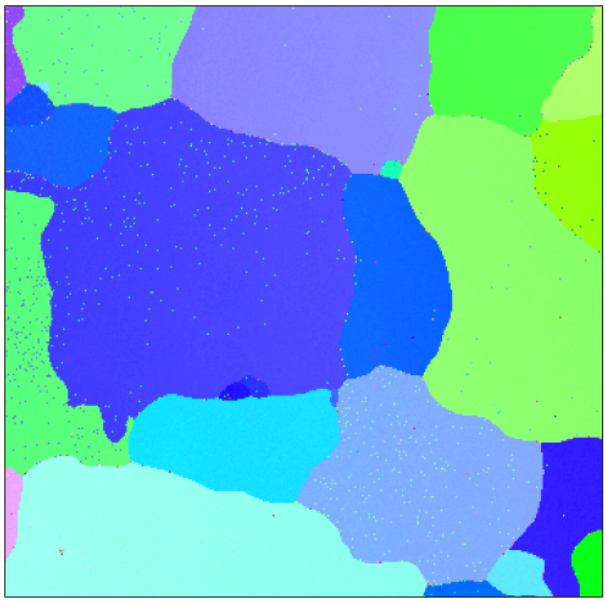
```
ipfKey = ipfTSLKey(ebsd(mineral_name));

%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;
```

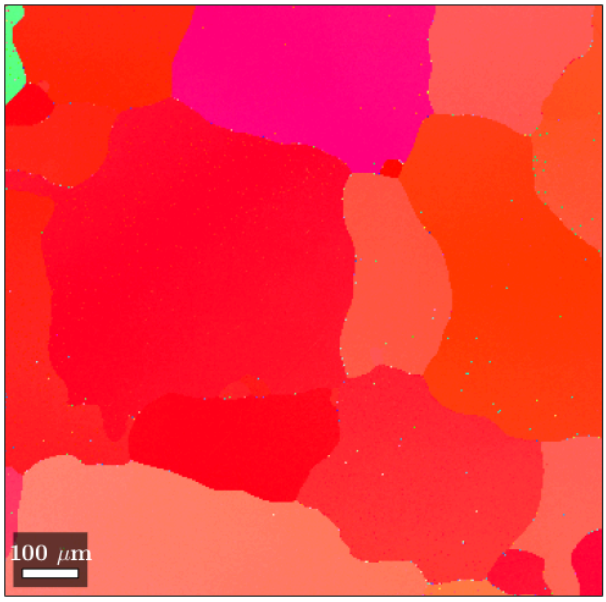
```
colors_Y = ipfKey.orientation2color(ebsd(mineral_name).orientations);
ipfKey.inversePoleFigureDirection = vector3d.Z;
colors_Z = ipfKey.orientation2color(ebsd(mineral_name).orientations);

%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');
```



6/mmm



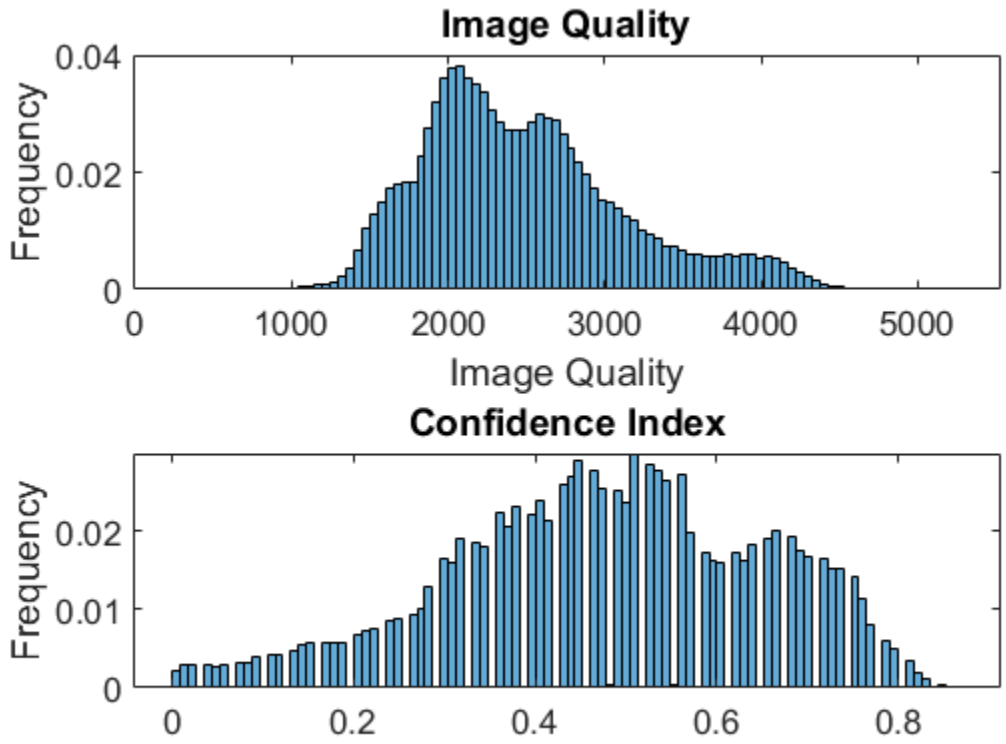




Plot a histogram of the quality data

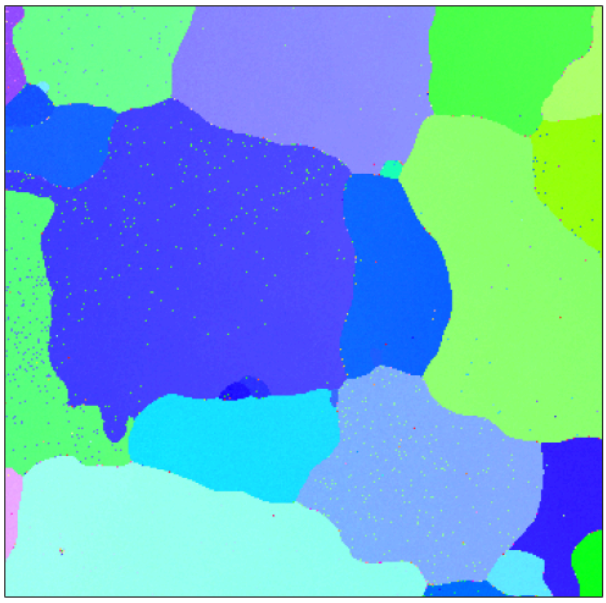
```
figure;
subplot(2,1,1);
h1=histogram(ebsd(mineral_name).iq,100);
h1.Normalization='probability';
title('Image Quality');
ylabel('Frequency');
xlabel('Image Quality');

subplot(2,1,2);
h2=histogram(ebsd(mineral_name).ci,100);
h2.Normalization='probability';
```

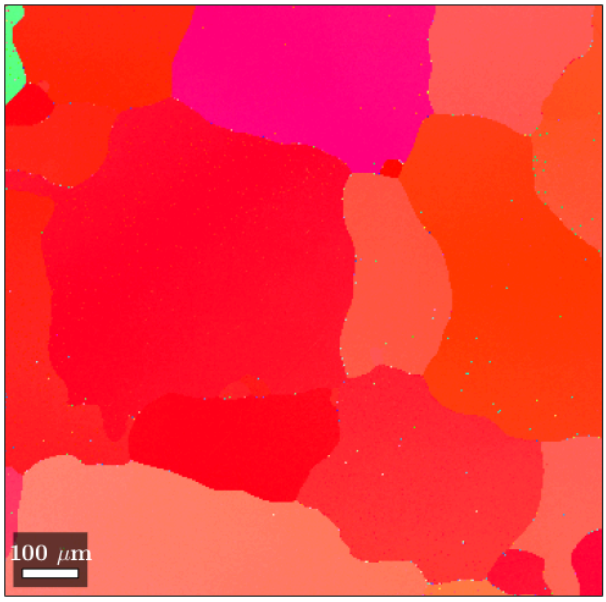
```
title('Confidence Index');
ylabel('Frequency');
```

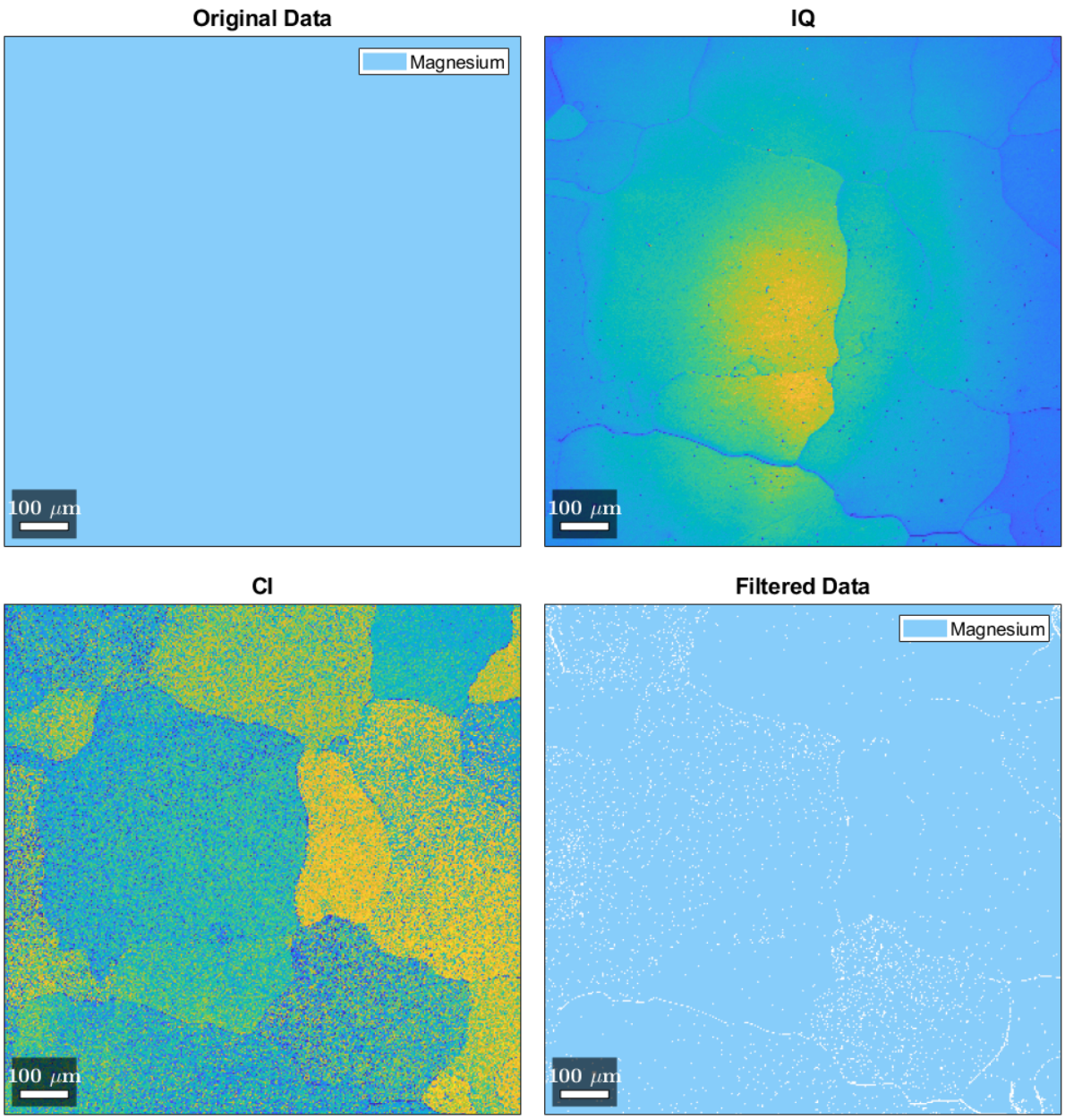


Filter the data based upon IQ and CI

```
%from these graphs I can select a threshold value for the phase to be ok
threshold_ci=0.1;
threshold_iq=1300;

%change the phase of the points we do not like to -1 [unindexed]
ebsd_filtered=ebsd;
ebsd_filtered(ebsd_filtered.iq < threshold_iq).phase=-1;
ebsd_filtered(ebsd_filtered.ci < threshold_ci).phase=-1;

figure;
plot(ebsd);
title('Original Data');
nextAxis;
plot(ebsd,ebsd.iq);
title('IQ');
nextAxis;
plot(ebsd,ebsd.ci);
title('CI');
nextAxis;
plot(ebsd_filtered);
title('Filtered Data');
```




100 μm



Clean up this data

personally I would rather see maps with data missing where it was misindexed

some people prefer that they do some filling in of data - this is data manipulation and so if you do not declare this you are committing scientific fraud

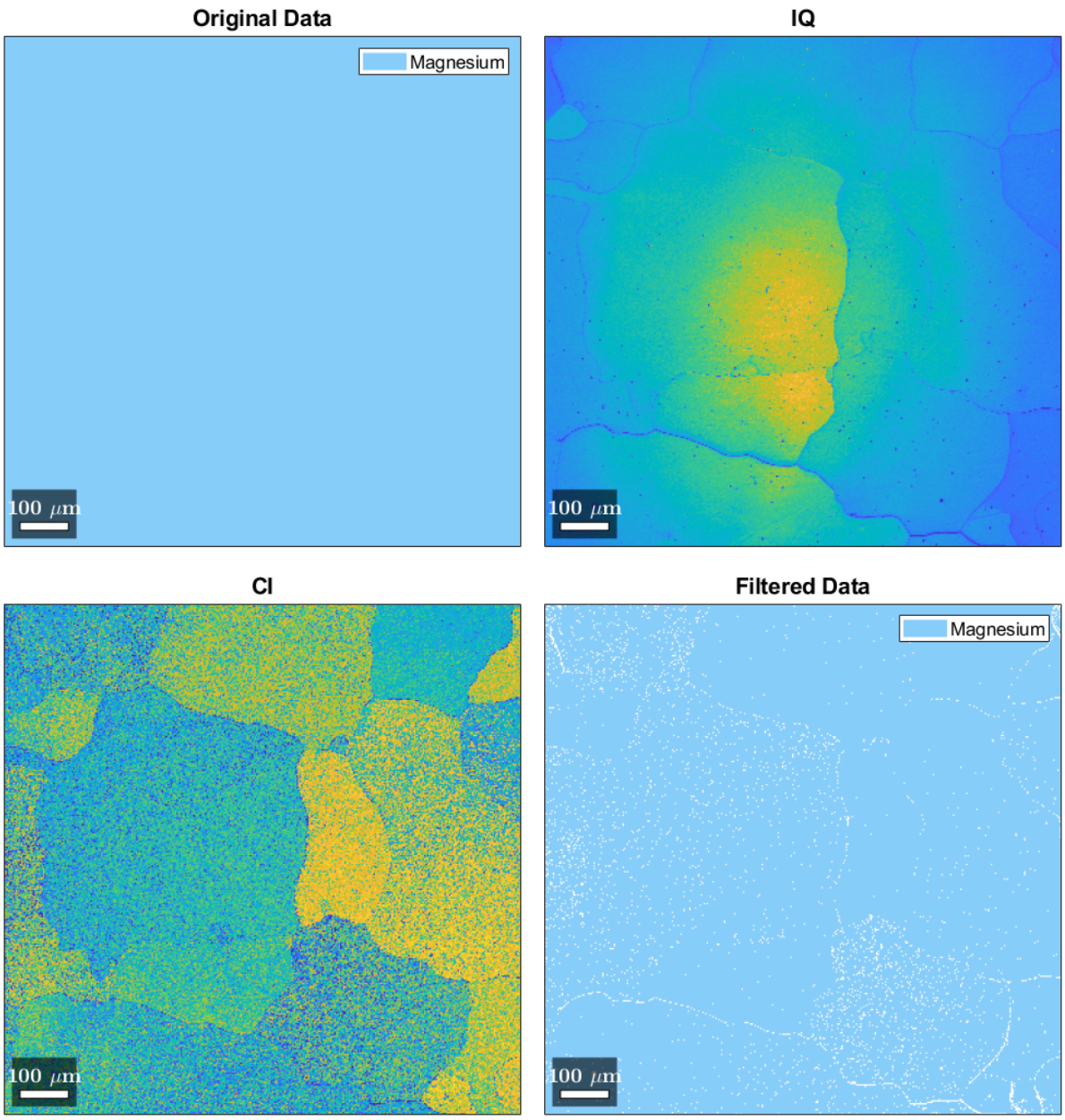
```
% grid this data
ebsd_filtered=gridify(ebsd_filtered);
ebsd_filtered_ok = ebsd_filtered('indexed');

%follow the method here: https://mtex-toolbox.github.io/EBSDFilling.html
```

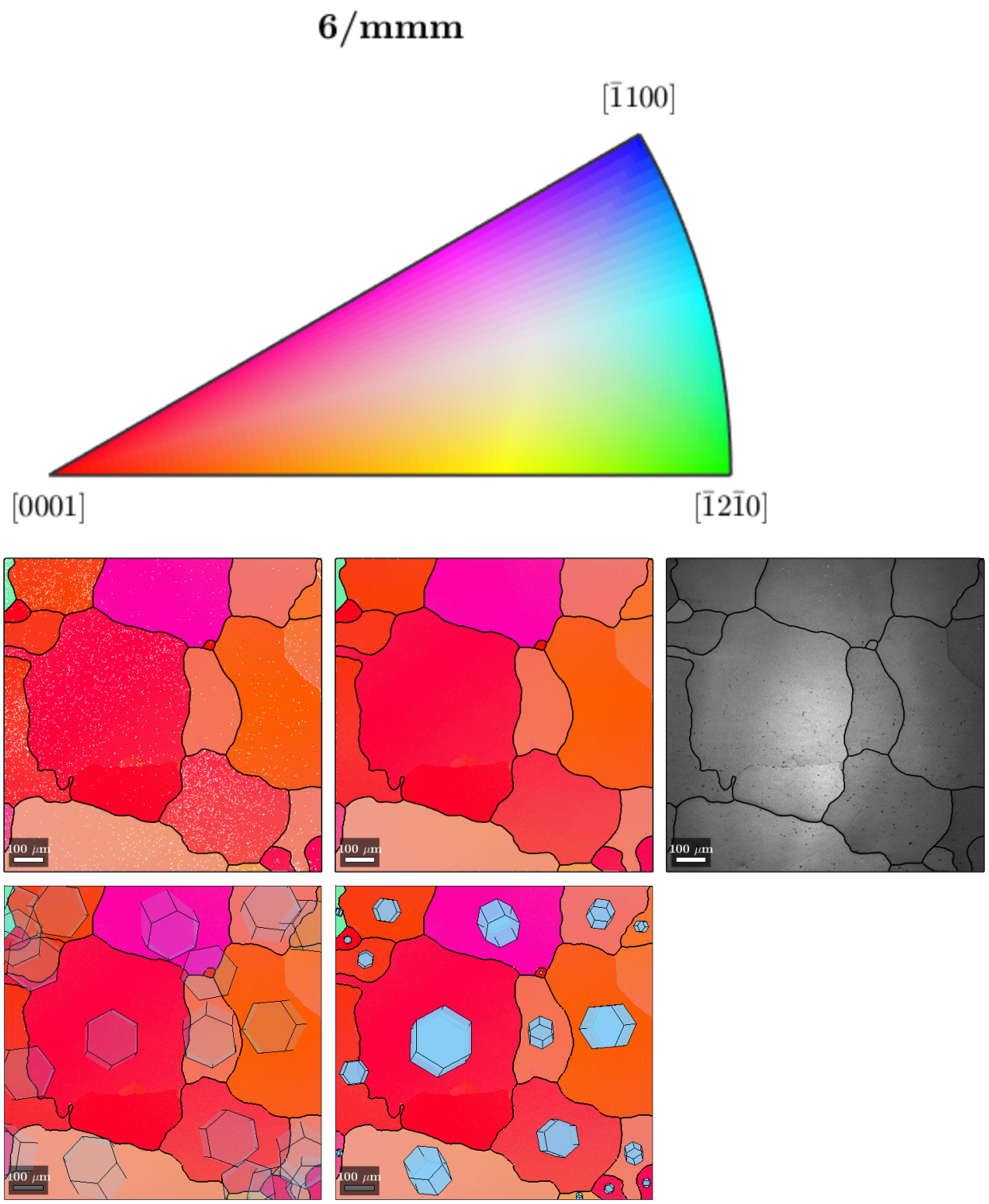
```
% reconstruct the grain structure
[grains,ebsd_filtered_ok.grainId,ebsd_filtered_ok.mis2mean] =
 calcGrains(ebsd_filtered_ok,'angle',10*degree);

% remove some very small grains
ebsd_filtered_ok(grains(grains.grainSize<5)) = [];

% redo grain segementation
[grains,ebsd_filtered_ok.grainId] =
 calcGrains(ebsd_filtered_ok,'angle',10*degree);

% smooth grain boundaries
grains = smooth(grains,5);

% fill in data
ebsd_filtered_ok_filled = fill(ebsd_filtered_ok,grains);
```

Plot the map with some unit cells on it

```
ipfKey = ipfHSVKey(ebsd_filtered_ok(mineral_name));
ipfKey2 = ipfHSVKey(ebsd_filtered_ok_filled(mineral_name));

f1=figure;
plot(ipfKey)

ipfKey.inversePoleFigureDirection = vector3d.Z;
colors_Z =
 ipfKey.orientation2color(ebsd_filtered_ok(mineral_name).orientations);
colors_Z2 =
 ipfKey2.orientation2color(ebsd_filtered_ok_filled(mineral_name).orientations);


figure;
% plot the orientation map
plot(ebsd_filtered_ok(mineral_name),colors_Z)

% and on top the grain boundaries
hold on
plot(grains.boundary,'linewidth',1.5)
hold off


nextAxis
plot(ebsd_filtered_ok_filled(mineral_name),colors_Z2)

% and on top the grain boundaries
hold on
plot(grains.boundary,'linewidth',1.5)
hold off

nextAxis
plot(ebsd,ebsd.iq)
```

```
% and on top the grain boundaries
hold on
plot(grains.boundary,'linewidth',1.5)
colormap('gray');
hold off

nextAxis
plot(ebsd_filtered_ok_filled(mineral_name),colors_Z2)

% and on top the grain boundaries
hold on
plot(grains.boundary,'linewidth',1.5)


% plot the unit cells
%plot the crystal shape for this point
scaling = 250; % 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(mineral_name).CS);
grain_centroids=grains.centroid;
plot(grain_centroids(:,1),grain_centroids(:,2),0, grains.meanOrientation * cS
 * scaling,'FaceAlpha',0.2); %make the hexagons also transparent so you can
 see the grains behind
xlim([min(ebsd.prop.x) max(ebsd.prop.x)]);
ylim([min(ebsd.prop.y) max(ebsd.prop.y)]);

%plot with the unit cells above, and scale their size by the grain size
nextAxis
plot(ebsd_filtered_ok_filled(mineral_name),colors_Z2)

% and on top the grain boundaries
hold on
plot(grains.boundary,'linewidth',1.5)
% plot the unit cells
%plot the crystal shape for this point
scaling = 250; % 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(mineral_name).CS);
grain_centroids=grains.centroid;
plot(grain_centroids(:,1),grain_centroids(:,2),-max(grains.area),
 grains.meanOrientation * cS .*sqrt(grains.area./4),'FaceAlpha',0.9); %make
 the hexagons also transparent so you can see the grains behind
xlim([min(ebsd.prop.x) max(ebsd.prop.x)]);
ylim([min(ebsd.prop.y) max(ebsd.prop.y)]);
```


Original Data
Magnesium
100 μm
IQ
100 μm
CI
100 μm
Filtered Data
Magnesium
100 μm

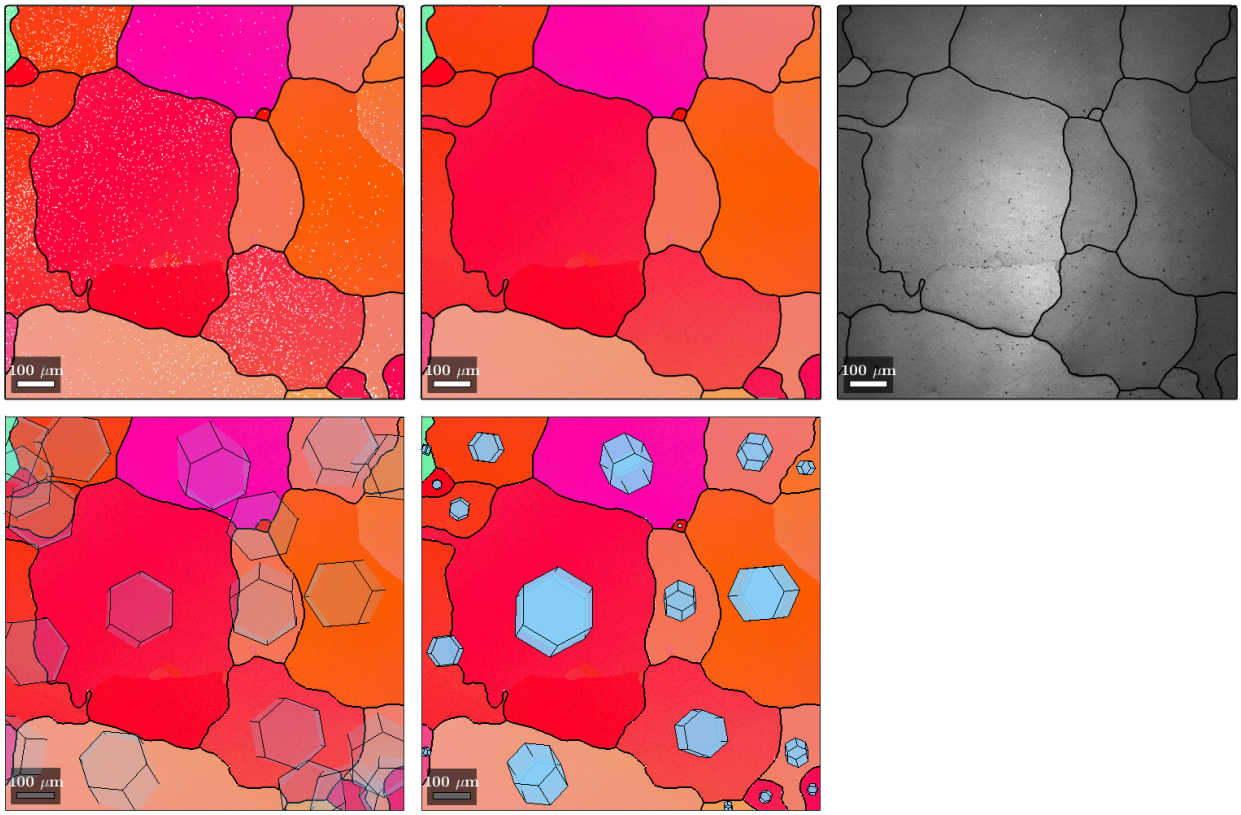


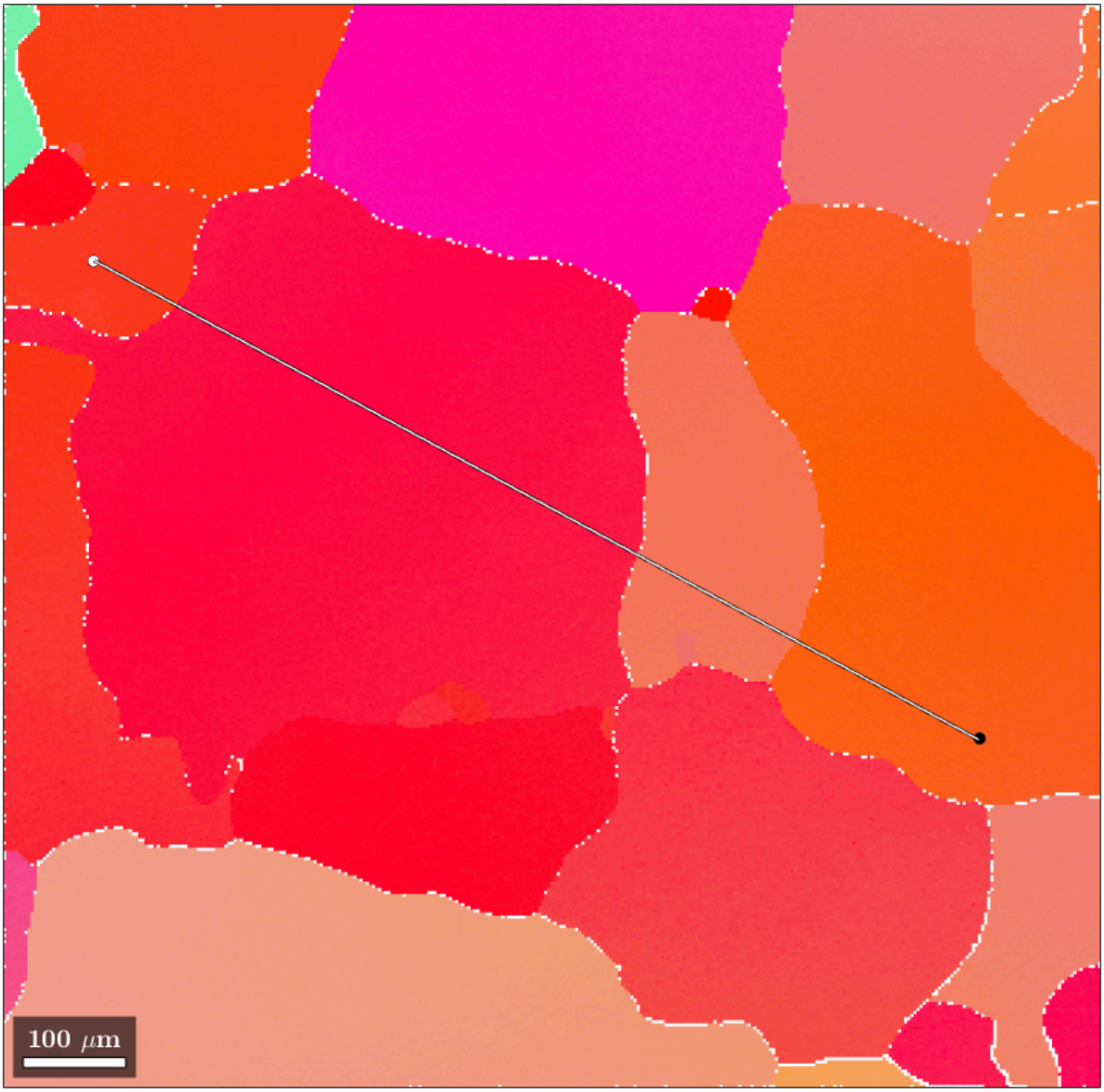
Measure the misorientation between two points

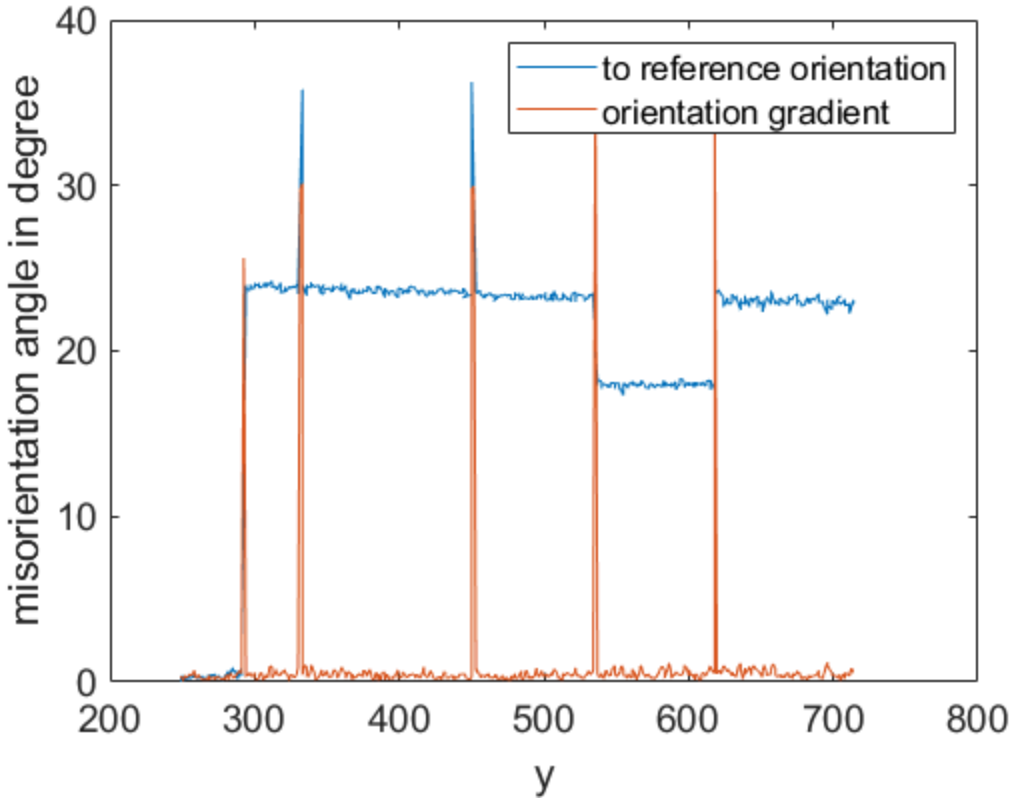
```
%first lets use the GUI tools to grab a few (well two) points
figure;
```

```
plot(ebsd_filtered_ok_filled(mineral_name),colors_Z2);
%click two points you want to extract
disp('click two points you want to measure the misorientations between')
[x_g,y_g]=ginput(2);

%find the pattern number for each of these points
pattern_number=zeros(size(x_g));
%find the nearest point in the data to this point
for n=1:numel(x_g)
    p_map=(ebsd.prop.x-x_g(n)).^2+(ebsd.prop.y-y_g(n)).^2;
    [mv,pattern_number(n)]=min(p_map);
end

ebsd_points=ebsd(pattern_number);

hold on;
scatter(ebsd_points.prop.x,ebsd_points.prop.y,'k');
scatter(ebsd_points(1).prop.x,ebsd_points(1).prop.y,'filled','w'); %make the
 first one white
scatter(ebsd_points(end).prop.x,ebsd_points(end).prop.y,'filled','k'); %make
 the last one black

%now we can calculate the misorientation angle between these two points
mis_angle_degree=angle(ebsd_points(1).orientations,ebsd_points(2).orientations)./
degree
mis_axis=axis(ebsd_points(1).orientations,ebsd_points(2).orientations)

% We can also use this line segment to calculate a profile
% https://mtex-toolbox.github.io/EBSDProfile.html
hold on;
line(x_g,y_g,'linewidth',2,'Color','k')
line(x_g,y_g,'linewidth',1,'Color','w')
lineSeg=[x_g,y_g];

ebsd_line = spatialProfile(ebsd,lineSeg);

figure;
% misorientation angle to the first orientation on the line
plot(ebsd_line.y,...
  angle(ebsd_line(1).orientations,ebsd_line.orientations)/degree)

% misorientation gradient
hold all
plot(0.5*(ebsd_line.y(1:end-1)+ebsd_line.y(2:end)),...
  angle(ebsd_line(1:end-1).orientations,ebsd_line(2:end).orientations)/degree)
hold off

xlabel('y'); ylabel('misorientation angle in degree')

legend('to reference orientation','orientation gradient')

click two points you want to measure the misorientations between

mis_angle_degree =
```

```
   22.9056

mis_axis = vector3d
         x          y          z
  0.150277 -0.562638  -0.81293
```




100 μm



Published with MATLAB® R2022b