Table of Contents


```
clear
home
close all

% Demonstration code for AstroEBSD for use at the 2019 RMS EBSD
 workshop
% CC-BY license https://creativecommons.org/licenses/by/4.0/
% created by Ben Britton and Alex Foden
%
% This code is best run 'step size' using 'run and advance'
%
% Ideally matlab 2018b+
% Requires installation of:
% global optimisation toolbox (for pattern centre searching)
% parallel computing toolbox (to speed up indexing of the map data)
%
% The Iron data set can be found here:
% https://doi.org/10.5281/zenodo.2609221
```

Toolbox locations for AstroEBSD and MTEX

```
location_astro='C:\Users\bbrit\Documents\GitHub\AstroEBSD\'; %Change
 this to your AstroEBSD location
location_mtex='C:\Users\bbrit\Documents\GitHub\mtex'; %Change this to
 where you keep your MTEX folder
```

run Astro and EBSD to start

```
run(fullfile(location_astro,'start_AstroEBSD.m'));
run(fullfile(location_mtex,'startup_mtex.m'));

Loading AstroEBSD 1.0  AstroEBSD file paths loaded
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>
```

Analyse a single pattern

```
pattern1=flipud(double(imread('pattern1.tif'))); %Make sure
 pattern1.tif is saved in your current working directory

% Normalise intensities
[EBSP_One.PatternIn,Settings_Cor ] = EBSP_BGCor( pattern1,[]);
Settings_Cor.radius=1;
Settings_Cor.radius_frac=0.95;

%build the phases
InputUser.Phase_Folder = fullfile(location_astro,'phases');
InputUser.Phase_Input  = {'Si'}; %Si, Ferrite
[ Crystal_UCell,Crystal_Family,Crystal_LUT,Settings_LUT,Phase_Num ] =
 Phase_Builder( InputUser.Phase_Input,InputUser.Phase_Folder );

%Set up the radon transform peak finder
Settings_Rad.theta_range=[-10 180 1]; %theta min, theta max, theta
 step - in degrees

%peak hunt
Settings_Rad.max_peaks=12; %max number of peaks to return
Settings_Rad.num_peak=20; %number of peaks to search for - peaks will
 be rejected
Settings_Rad.theta_search_pix=6; %search size in theta steps
Settings_Rad.rho_search_per=0.2; %radon search in fractions
Settings_Rad.min_peak_width=0.002; %min rseperation of the peak width,
 in pixels
```

Load in the GUI

```
%play with the GUI
%Close the GUI to have the code continue automatically
```

```
Settings_PCin.start=[0.5,0.3,0.6]; %[PCx, PCy, PCz] using Bruker
 conventions

Astro_EBSPset(EBSP_One,Settings_Cor,Settings_Rad,Settings_PCin,InputUser);
```

AstroEBSD GUI is active
Close the window or use CRTL + C to return to the command line

Do the indexing offline

```
%This is the example
EBSP_One.PC=[0.5,0.3,0.6]; %[PCx, PCy, PCz] using Bruker conventions

%Radon transform
[ EBSP_One.Peak_Centre,EBSP_One.Single.Peak_Set_All,EBSP_One.Peak_Set_All,...

 EBSP_One.R_EBSP,EBSP_One.R_Edge,EBSP_One.R_rho,EBSP_One.R_theta ] ...
            = EBSP_RadHunt( EBSP_One.PatternIn,Settings_Rad);

% Convert the bands to normal space
[ EBSP_One.nhat_gnom] =
 EBSP_NormConv( EBSP_One.Peak_Centre,size(EBSP_One.PatternIn),EBSP_One.PC);

% Index
[EBSP_One.rotdata,EBSP_One.banddata]=EBSP_Index(EBSP_One.nhat_gnom,Crystal_LUT{1},
```

Now read a H5 file

```
InputUser.HDF5_folder='C:\Users\bbrit\Documents\EBSD'; %Change this to
 the file location in whch you have saved the example data
InputUser.HDF5_file='Demo_Ben_16bin.h5';
InputUser.Phase_Input  = {'Ferrite'};
Settings_PCin.start=[0.5010 0.4510 0.5870]; %Fe

%Read the h5 data
[ MapData,MicroscopeData,PhaseData,EBSPData ] =
 bReadHDF5( InputUser );

%Read an EBSP
[ EBSP_FeR ] = bReadEBSP(EBSPData,1);
EBSP_Fe.PatternIn=EBSP_FeR;

%background correction settings

%gaussian flatten (removes the low frequency bg)
Settings_Cor_Fe.gfilt=1; %use a low pass filter (do you mean high
 pass?)
Settings_Cor_Fe.gfilt_s=4; %low pass filter sigma

%radius mask (crops to a circle)
Settings_Cor_Fe.radius=1; %use a radius mask
Settings_Cor_Fe.radius_frac=0.98; %fraction of the pattern width to
 use as the mask
```

```
%split BG fix (removes vertical seam)
Settings_Cor_Fe.SplitBG=1;

%inspect the pattern & give settings back to the user
[Settings_Corout,Settings_Radout,Settings_PCout]=Astro_EBSPset(EBSP_Fe,Settings_Co
```

```
Warning: SE Image not loaded
Warning: Coordinate systems not loaded
Warning: Phase data not loaded
AstroEBSD GUI is active
Close the window or use CRTL + C to return to the command line
```

Read the EBSD patterns into the RAM - this takes 0.5GB of RAM, but makes every thing else easier

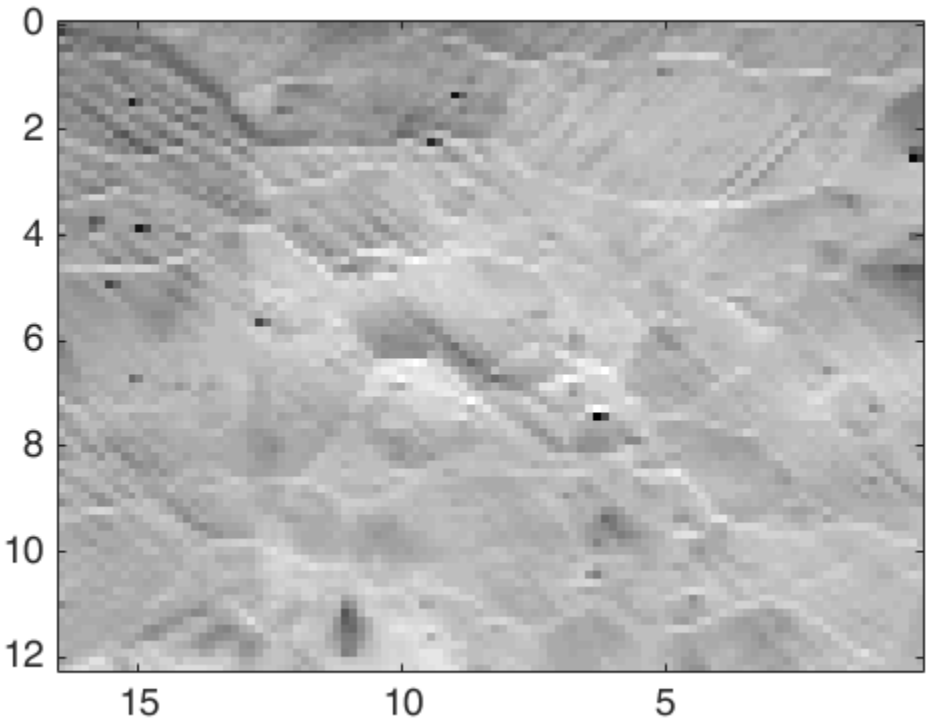
```
EBSP_all=zeros([size(EBSP_FeR) MicroscopeData.NPoints]);
for p=1:MicroscopeData.NPoints
    [ EBSP_all(:,:,p) ] = bReadEBSP(EBSPData,p); %This can take a
 minute or two so be patient if it doesn't seem to work straighht away

    %provide some feedback
    if 1000*round(p/1000) == p
        disp(['Pattern ' int2str(p) ' of '
 int2str(MicroscopeData.NPoints) ' patterns corrected']);
    end
end
```

```
Pattern 1000 of 9130 patterns corrected
Pattern 2000 of 9130 patterns corrected
Pattern 3000 of 9130 patterns corrected
Pattern 4000 of 9130 patterns corrected
Pattern 5000 of 9130 patterns corrected
Pattern 6000 of 9130 patterns corrected
Pattern 7000 of 9130 patterns corrected
Pattern 8000 of 9130 patterns corrected
Pattern 9000 of 9130 patterns corrected
```

Now we can start playing

```
EBSP_sum=zeros(MicroscopeData.NPoints,1);
for p=1:MicroscopeData.NPoints
    EBSP_sum(p)=sum(EBSP_all(:,:,p),'all');
% EBSP_sum(p)=sum(sum(EBSP_all(:,:,p))); % There are two versions
 of this here because of a bug fix, R2018b onwards can use the first
 verion, earlier versions need the second
end

%sort this into a 2D Map
[Map_EBSP_sum] = bMapSort(MapData,MicroscopeData,EBSP_sum);
```

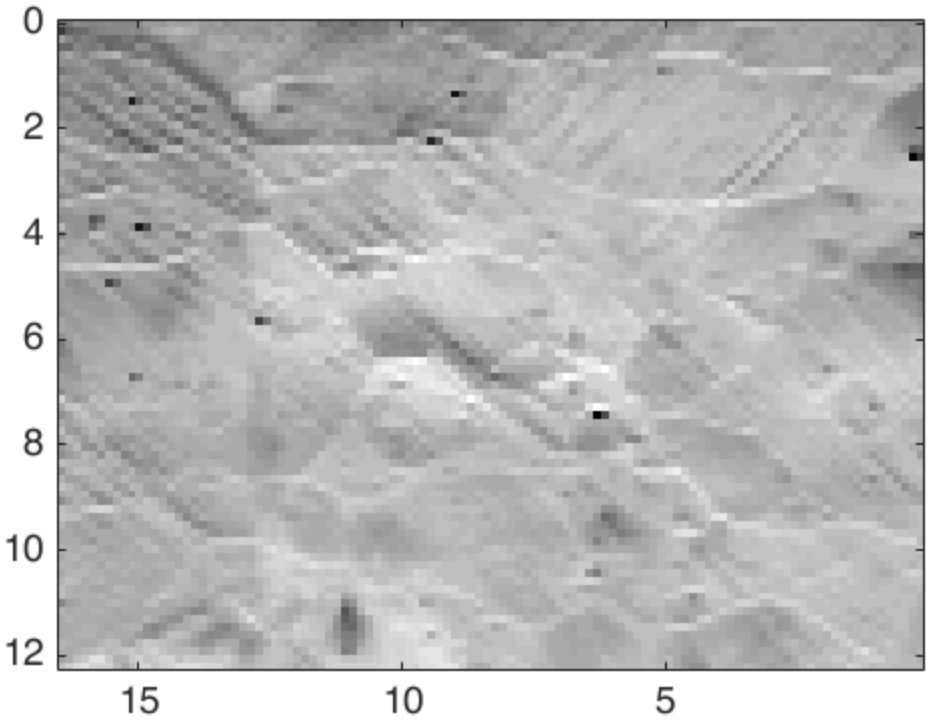
```
%Map the EBSD scanning coords
Map_XSample=bMapSort(MapData,MicroscopeData,MapData.XSample);
Map_YSample=bMapSort(MapData,MicroscopeData,MapData.YSample);

%plot the secondary electron map
figure;
sp1=subplot(1,1,1); %creates an axis
imagesc(Map_XSample(1,:),Map_YSample(:,1)',Map_EBSP_sum); %plots the
 data
axis image; axis tight; colormap('gray'); axis ij;
 sp1.XDir='reverse'; %sets the axis to how they are expected
```

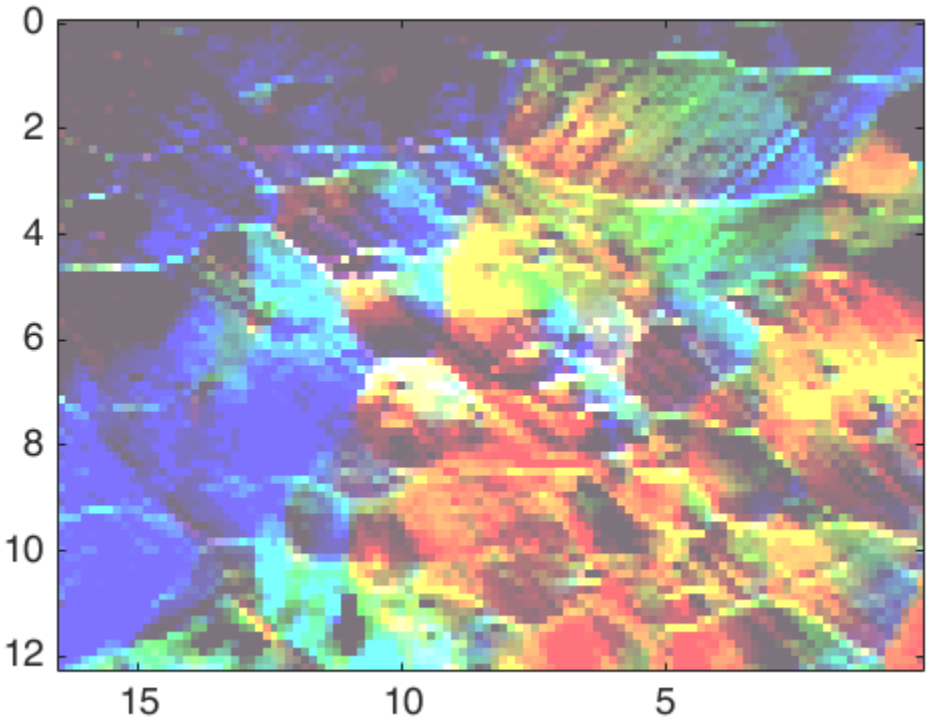


plot a dummy argus image

```
%create zeros
EBSP_sumR=zeros(MicroscopeData.NPoints,1);
EBSP_sumG=zeros(MicroscopeData.NPoints,1);
EBSP_sumB=zeros(MicroscopeData.NPoints,1);

%sum three horizontal strips from the EBSP
for p=1:MicroscopeData.NPoints
    EBSP_sumR(p)=sum(EBSP_all(:,1:33,p),'all');
    EBSP_sumG(p)=sum(EBSP_all(:,34:66,p),'all');
    EBSP_sumB(p)=sum(EBSP_all(:,66:100,p),'all');
```

```
%     EBSP_sumR(p)=sum(sum(EBSP_all(:,1:33,p))); %There are two
 versions of this because of the same versioning issue earlier
%     EBSP_sumG(p)=sum(sum(EBSP_all(:,34:66,p)));
%     EBSP_sumB(p)=sum(sum(EBSP_all(:,66:100,p)));
end

%convert into maps
[Map_EBSP_sumR] = bMapSort(MapData,MicroscopeData,EBSP_sumR);
[Map_EBSP_sumG] = bMapSort(MapData,MicroscopeData,EBSP_sumG);
[Map_EBSP_sumB] = bMapSort(MapData,MicroscopeData,EBSP_sumB);

%normalise the channels
%zero mean
Map_EBSP_sumR=Map_EBSP_sumR-mean(Map_EBSP_sumR(:));
Map_EBSP_sumG=Map_EBSP_sumG-mean(Map_EBSP_sumG(:));
Map_EBSP_sumB=Map_EBSP_sumB-mean(Map_EBSP_sumB(:));

%fix the STDev
Map_EBSP_sumR=Map_EBSP_sumR./std(Map_EBSP_sumR(:));
Map_EBSP_sumG=Map_EBSP_sumG./std(Map_EBSP_sumG(:));
Map_EBSP_sumB=Map_EBSP_sumB./std(Map_EBSP_sumB(:));

%normalise the histogram channels
Map_EBSP_sumR_eq=histeq(Map_EBSP_sumR);
Map_EBSP_sumG_eq=histeq(Map_EBSP_sumG);
Map_EBSP_sumB_eq=histeq(Map_EBSP_sumB);

%put into a RGB array
Map_EBSP_sum_RGB=Map_EBSP_sumR_eq;
Map_EBSP_sum_RGB(:,:,2)=Map_EBSP_sumG_eq;
Map_EBSP_sum_RGB(:,:,3)=Map_EBSP_sumB_eq;

%plot the final figure
figure;
sp1=subplot(1,1,1); %creates an axis
image(Map_XSample(1,:),Map_YSample(:,1)',Map_EBSP_sum_RGB);
 axis image;
axis image; axis tight;  axis ij; sp1.XDir='reverse'; %sets the axis
 to how they are expected
```


0
2
4
6
8
10
12
15
10
5


0
2
4
6
8
10
12
15
10
5

Now process the patterns

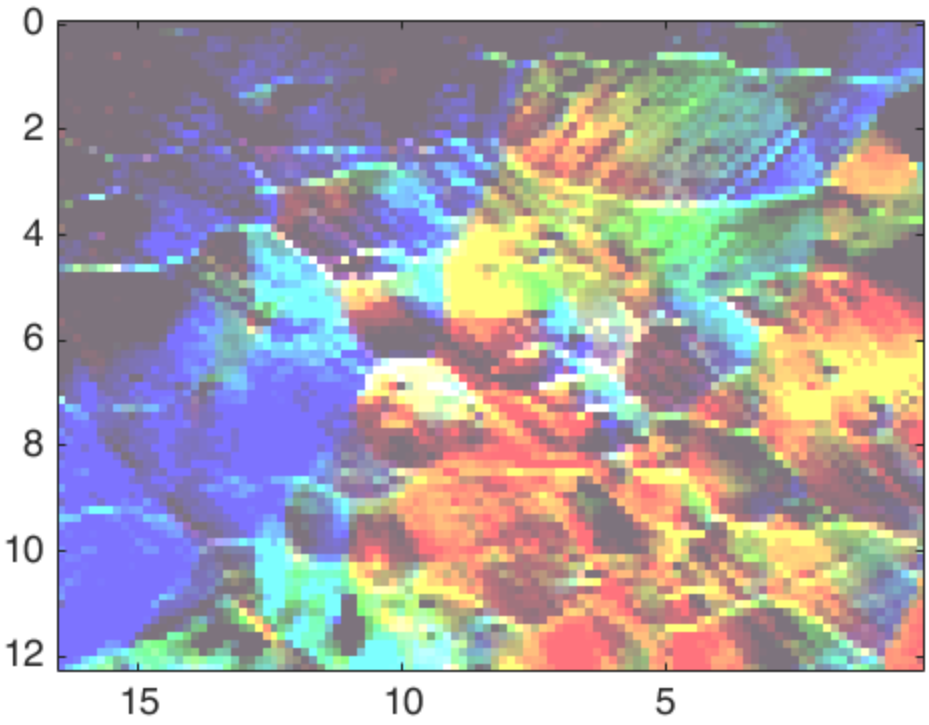
```
EBSP_BG=zeros([size(EBSP_FeR) MicroscopeData.NPoints]);

for p=1:MicroscopeData.NPoints
    [EBSP_BG(:,:,p),~ ] =
 EBSP_BGCor( EBSP_all(:,:,p),Settings_Cor); %Background correct the
 patterns

        %provide some feedback
    if 1000*round(p/1000) == p
        disp(['Pattern ' int2str(p) ' of '
 int2str(MicroscopeData.NPoints) ' patterns corrected']);
    end
end
```

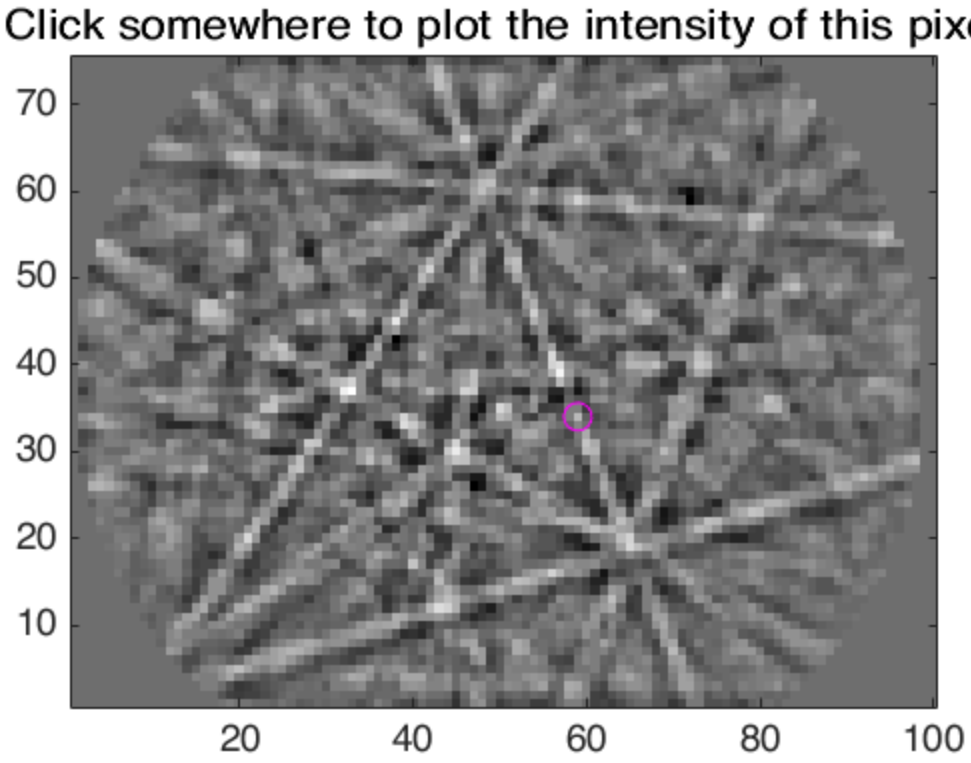
```
Pattern 1000 of 9130 patterns corrected
Pattern 2000 of 9130 patterns corrected
Pattern 3000 of 9130 patterns corrected
Pattern 4000 of 9130 patterns corrected
Pattern 5000 of 9130 patterns corrected
Pattern 6000 of 9130 patterns corrected
Pattern 7000 of 9130 patterns corrected
Pattern 8000 of 9130 patterns corrected
Pattern 9000 of 9130 patterns corrected
```

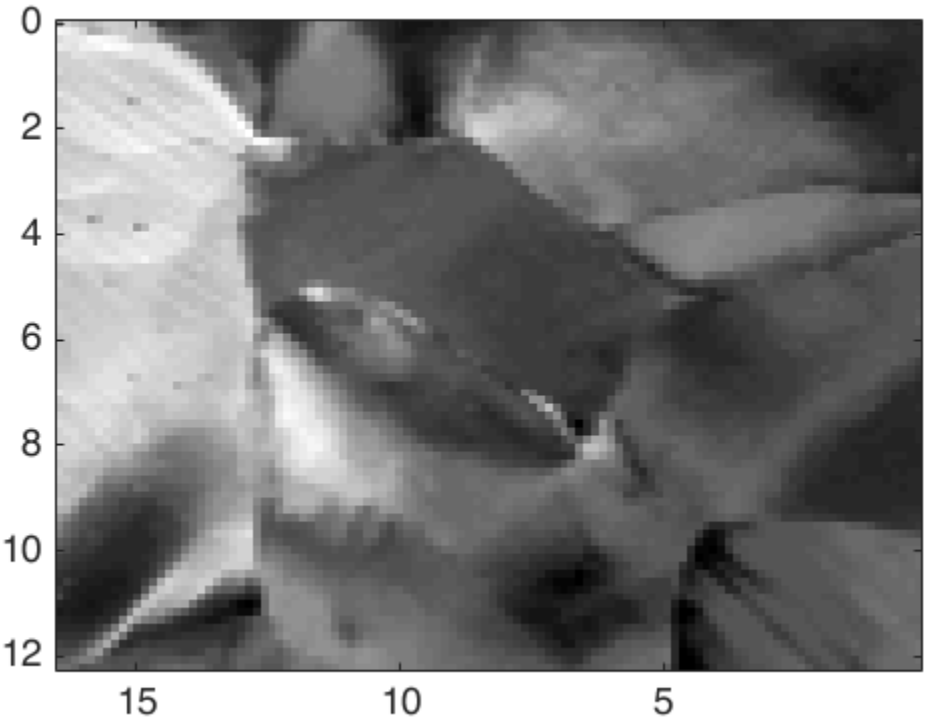
Extract data from 1 pixel

```
figure;
imagesc(EBSP_BG(:,:,1)); axis image; axis xy; title('Click somewhere
 to plot the intensity of this pixel');
colormap('gray')
%take a mouse input
[x,y]=ginput(1);
%round to a whole pixel location
x=round(x); y=round(y);
%plot this location
hold on; scatter(x,y,'m');

%extract this pixel from the EBSP data
EBSP_pixel=EBSP_BG(y,x,:);
%shift the dimensions from (1,1,NUMPTS) to (NUMPTS,1);
EBSP_pixel=shiftdim(EBSP_pixel,2);

%turn into a map
Map_EBSP_pixel=bMapSort(MapData,MicroscopeData,EBSP_pixel);

%plot the map
figure;
sp1=subplot(1,1,1); %creates an axis
imagesc(Map_XSample(1,:),Map_YSample(:,1)',Map_EBSP_pixel); %plots the
 data
```

```
axis image; axis tight; colormap('gray'); axis ij;
 sp1.XDir='reverse'; %sets the axis to how they are expected
```

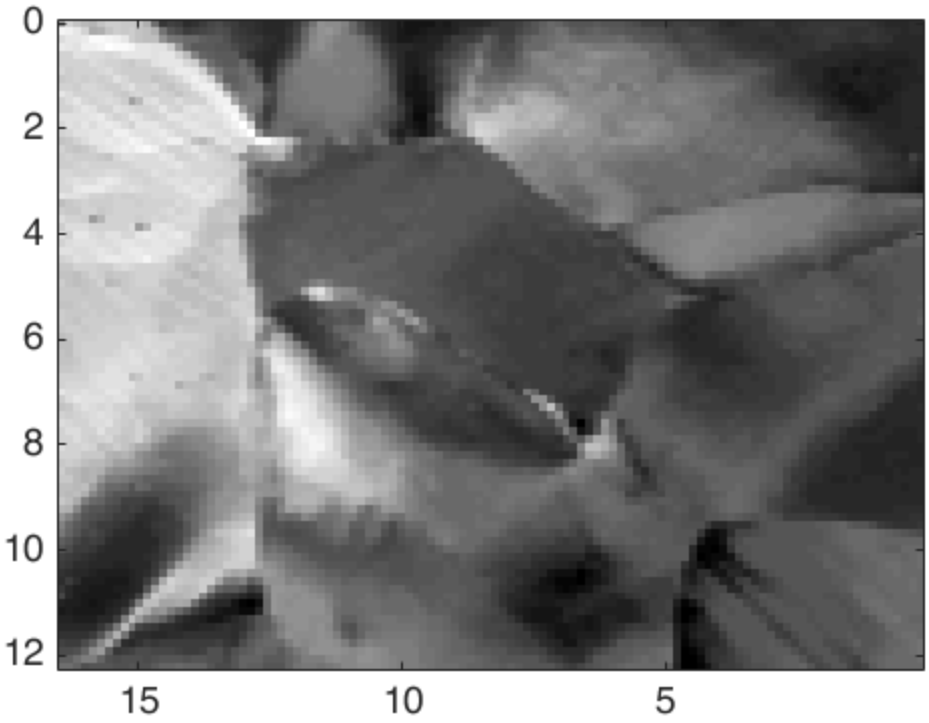


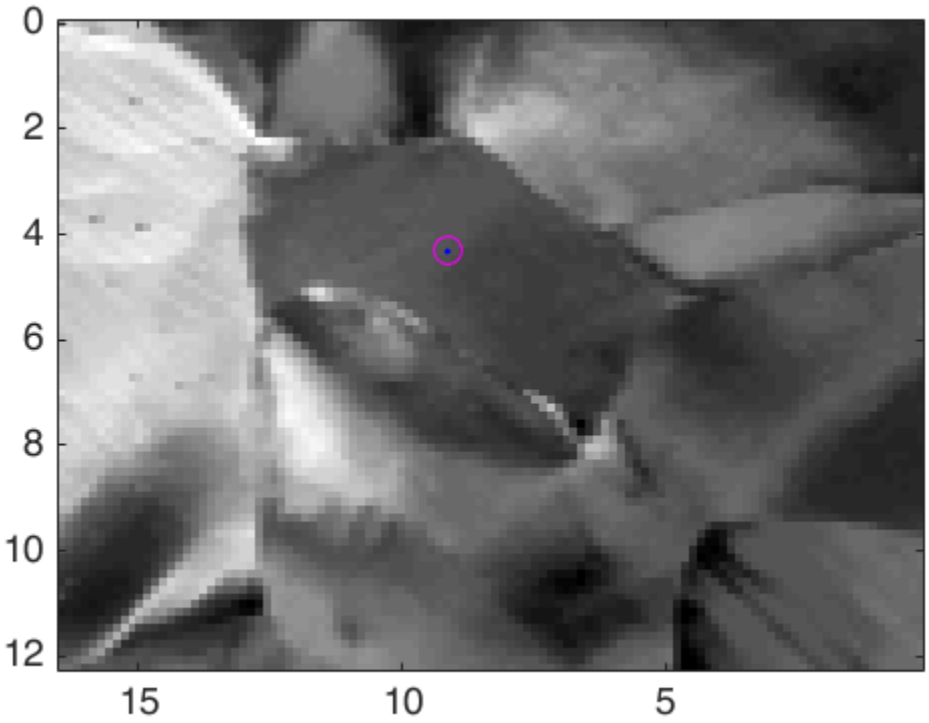

Click somewhere to plot the intensity of this pixel
70
60
50
40
30
20
10
20
40
60
80
100

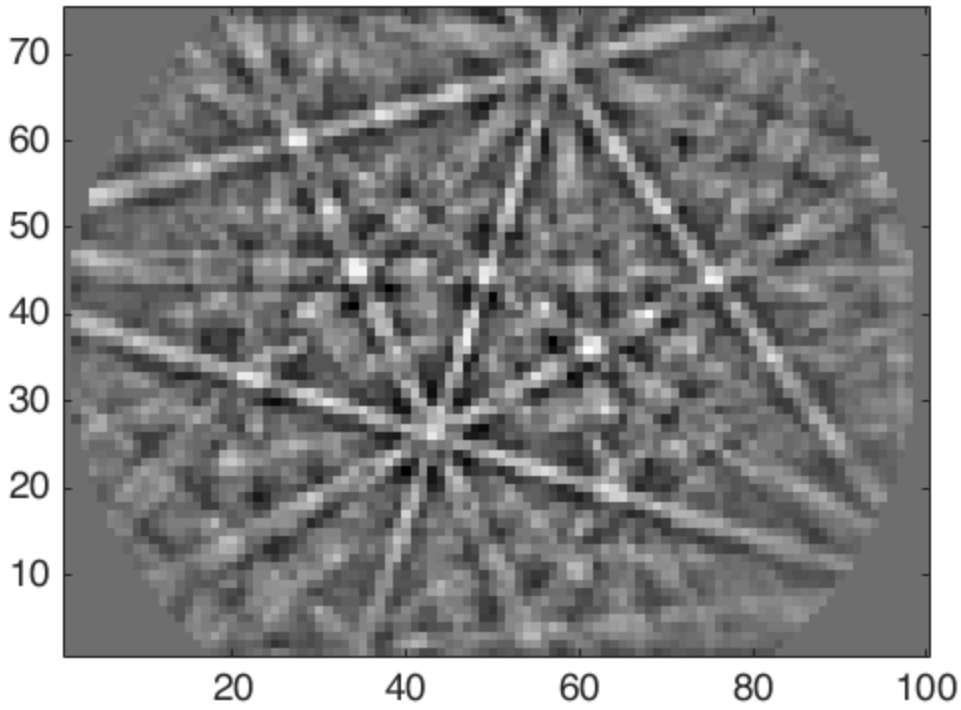

0
2
4
6
8
10
12
15
10
5

From a map, plot the EBSP

```
figure; %crete new figure
sp1=subplot(1,1,1); %creates an axis
imagesc(Map_XSample(1,:),Map_YSample(:,1)',Map_EBSP_pixel); %plots the
 data
axis image; axis tight; colormap('gray'); axis ij;
 sp1.XDir='reverse'; %sets the axis to how they are expected
[EBSP_x,EBSP_y]=ginput(1); %one mouse input
hold on; scatter(EBSP_x,EBSP_y,'m'); %plot this point

%calculate the pythagorian distance
Map_Distance_XY=(Map_XSample-EBSP_x).^2+(Map_YSample-EBSP_y).^2;

%calculate the minimum from a 2D array
[~,ii]=min(Map_Distance_XY(:));
[IY,IX]=ind2sub(size(Map_Distance_XY),ii);

%plot back on the image, to show which pixel is being selected
XI=Map_XSample(IY,IX);
YI=Map_YSample(IY,IX);
scatter(XI,YI,5,'b','filled')

%convert the pattern mapping into a 2D array to index
Map_PNum=bMapSort(MapData,MicroscopeData,MapData.PMap);

%index this map for that pattern
pattern_number=Map_PNum(IY,IX);

%plot this pattern
figure; %crete new figure
sp1=subplot(1,1,1); %creates an axis
imagesc(EBSP_BG(:,:,pattern_number)); axis image; axis xy; axis tight;
 colormap('gray');
```


0
2
4
6
8
10
12
15
10
5

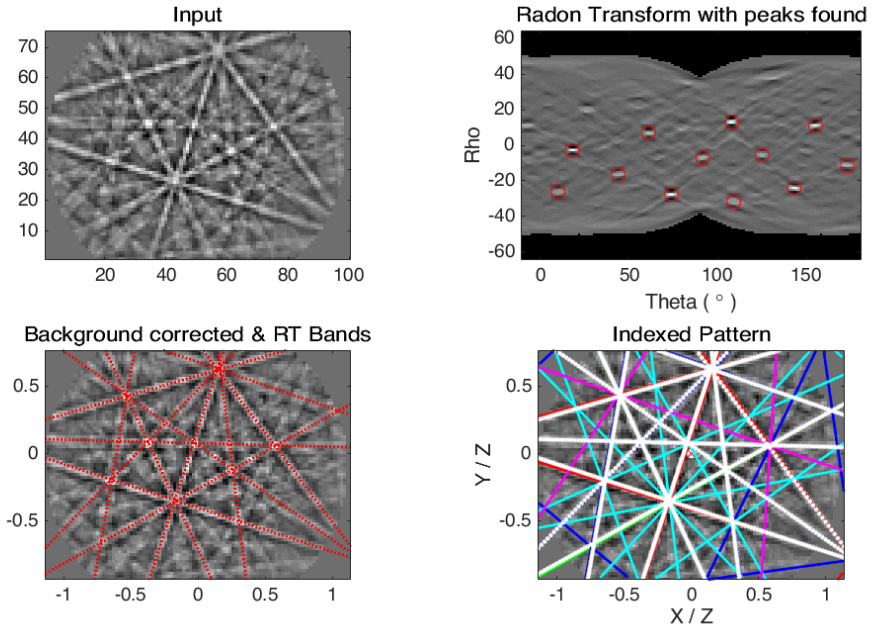

0
2
4
6
8
10
12
15
10
5



Use the astro settings as before to find the PC & index

```
EBSP_Fe_One.PatternIn=EBSP_BG(:,:,pattern_number);
EBSP_Fe_One.PC=[0.5010 0.4510 0.5870]; %Fe

InputUser.Phase_Folder = fullfile(location_astro,'phases');
InputUser.Phase_Input  = {'Ferrite'};
[ Crystal_UCell,Crystal_Family,Crystal_LUT,Settings_LUT,Phase_Num ] =
 Phase_Builder( InputUser.Phase_Input,InputUser.Phase_Folder );

%Radon transform
[ EBSP_Fe_One.Peak_Centre,EBSP_Fe_One.Single.Peak_Set_All,EBSP_Fe_One.Peak_Set_All

 EBSP_Fe_One.R_EBSP,EBSP_Fe_One.R_Edge,EBSP_Fe_One.R_rho,EBSP_Fe_One.R_theta ] ...
            = EBSP_RadHunt( EBSP_Fe_One.PatternIn,Settings_Rad);

% Convert the bands to normal space
[ EBSP_Fe_One.nhat_gnom] =
 EBSP_NormConv( EBSP_Fe_One.Peak_Centre,size(EBSP_Fe_One.PatternIn),EBSP_Fe_One.PC

R_x=@(theta)[1 0 0;0 cos(theta) sin(theta);0 -sin(theta) cos(theta)];
rot_det=R_x(MicroscopeData.TotalTilt);
```

```
% Index this pattern
[EBSP_Fe_One.rotdata{1},EBSP_Fe_One.banddata{1}]=EBSP_Index(EBSP_Fe_One.nhat_gnom,

%generate the geometry
[ EBSP_Fe_One.PatternGeometry ] =
 EBSP_Gnom( Settings_Cor,EBSP_Fe_One.PC );

EBSP_OneFigure=Plot_SinglePattern(EBSP_Fe_One,Crystal_UCell,Crystal_LUT,1);
```

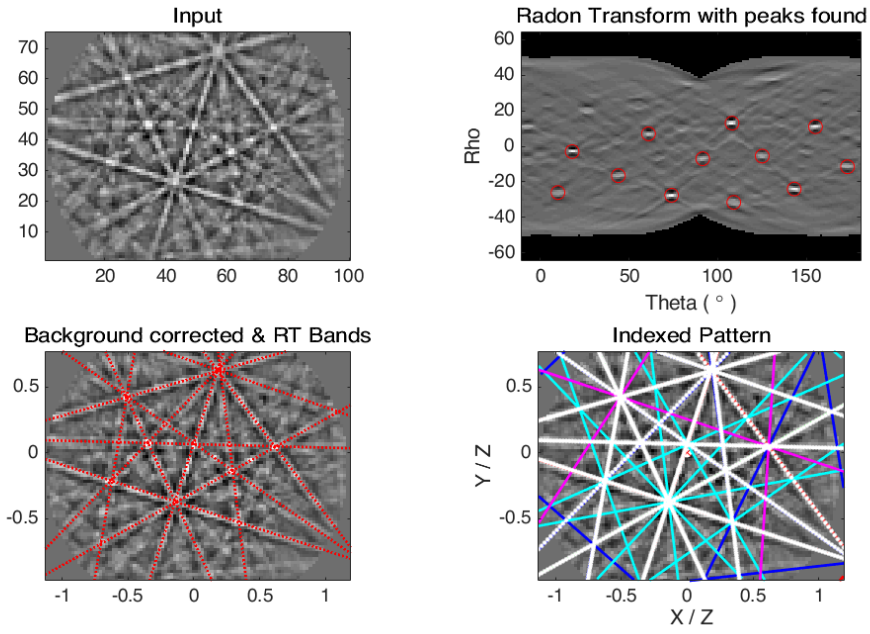


Find the pattern centre for one pattern

```
%For this to work you need to have the optimization and global
 optimization
%toolboxes installed

EBSP_Fe_PC.PatternIn=EBSP_Fe_One.PatternIn;

%copy over the PC found earlier
Settings_PCin.start=EBSP_Fe_One.PC;

%find the pattern centre
[EBSP_Fe_PCOut] =
 EBSP_PCSearch(EBSP_Fe_PC,Settings_Cor,Settings_Rad,Settings_PCin,Phase_Num,Crysta

%generate the geometry
[ EBSP_Fe_PCOut.PatternGeometry ] =
 EBSP_Gnom( Settings_Cor,EBSP_Fe_PCOut.PC );
%index the pattern
[ EBSP_Fe_PCOut.nhat_gnom] =
 EBSP_NormConv( EBSP_Fe_PCOut.Peak_Centre,size(EBSP_Fe_PCOut.PatternIn),EBSP_Fe_PC
[EBSP_Fe_PCOut.rotdata{1},EBSP_Fe_PCOut.banddata{1}]=EBSP_Index(EBSP_Fe_PCOut.nhat
```

```
%plot the result
EBSP_OneFigure=Plot_SinglePattern(EBSP_Fe_PCOut,Crystal_UCell,Crystal_LUT,1);
```

```
                                    Best                Mean       Stall
Generation         Func-count       f(x)                f(x)       Generations
     1                 60          0.006575            0.6689          0
     2                 90           0.00513            0.4503          0
     3                120           0.00513            0.3601          1
     4                150           0.00513            0.2094          2
     5                180           0.00513            0.1859          3
     6                210           0.00513           0.05486          4
     7                240           0.00513            0.1955          5
     8                270           0.00513            0.1731          6
     9                300           0.00513           0.09184          7
    10                330          0.004819            0.1067          0
    11                360          0.004819           0.03741          1
    12                390          0.004819           0.03181          2
    13                420          0.004557           0.03098          0
    14                450          0.004557          0.005277          1
    15                480          0.004557          0.005227          2
Optimization terminated: maximum number of generations exceeded.
```



Do this for a whole map

```
%convert the input data into a map
disp('Converting EBSD data to a map');
[Data_InputMap] = EBSD_Map(MapData,MicroscopeData);
```

```
%radon transform (from EBSP data on disk - this uses original
 AstroEBSD codes)
%it will use the parallel compute toolbox - you can adjust this if you
 do not have access

disp('Perfoming Radon transform on the whole map');
[Peak_Centres,Peak_Quality,Peak_NBands,EBSD_Info ] =
 Map_Radon( Data_InputMap,EBSPData,Settings_Cor,Settings_Rad );

disp('Searching for the pattern centre')
Settings_PCin.start=[0.493 0.452 0.569]; %Fe
Settings_PCin.array=[5 5]; %[#X,#Y] points extracted from map - will
 fit a PC to these points & then fit a plane
Settings_PCin.range=[0.15 0.15 0.15]; %+- these values

%find a pattern centre model
[PCOut] =
 Map_PCSearch(Data_InputMap,Peak_Centres,Peak_NBands,EBSD_Info,Crystal_LUT,Crystal

disp('Indexing the map')
%index
[Indexed_Rotdata,Indexed_Banddata] = ...

 Map_Index( Data_InputMap,Peak_Centres,Peak_NBands,Phase_Num,PCOut.Fit_2nd,Crystal

Converting EBSD data to a map
Perfoming Radon transform on the whole map
Starting parallel pool (parpool) using the 'local' profile ...
connected to 4 workers.
Searching for the pattern centre
Indexing the map
```

Write data to a h5 file

```
%create the file container
OutputUser=InputUser;

OutputUser.HDF5_file=[OutputUser.HDF5_file(1:end-3) '_Astro.h5'];
%we assume that this is an h5 file, hdf5 will cause issues with the
 (end-3)
OutputUser.DataName=InputUser.HDF5_file(1:end-3);
OutputUser.HDF5FullFile=fullfile(OutputUser.HDF5_folder,OutputUser.HDF5_file);

dtype='/EBSD/Data/'; %EBSD data location
htype='/EBSD/Header/'; %Header data location (e.g. microscope
 settings)

%Pattern centre
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'DD',PCOut.Fit_2nd.
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'PCX',PCOut.Fit_2nd
 the data
```

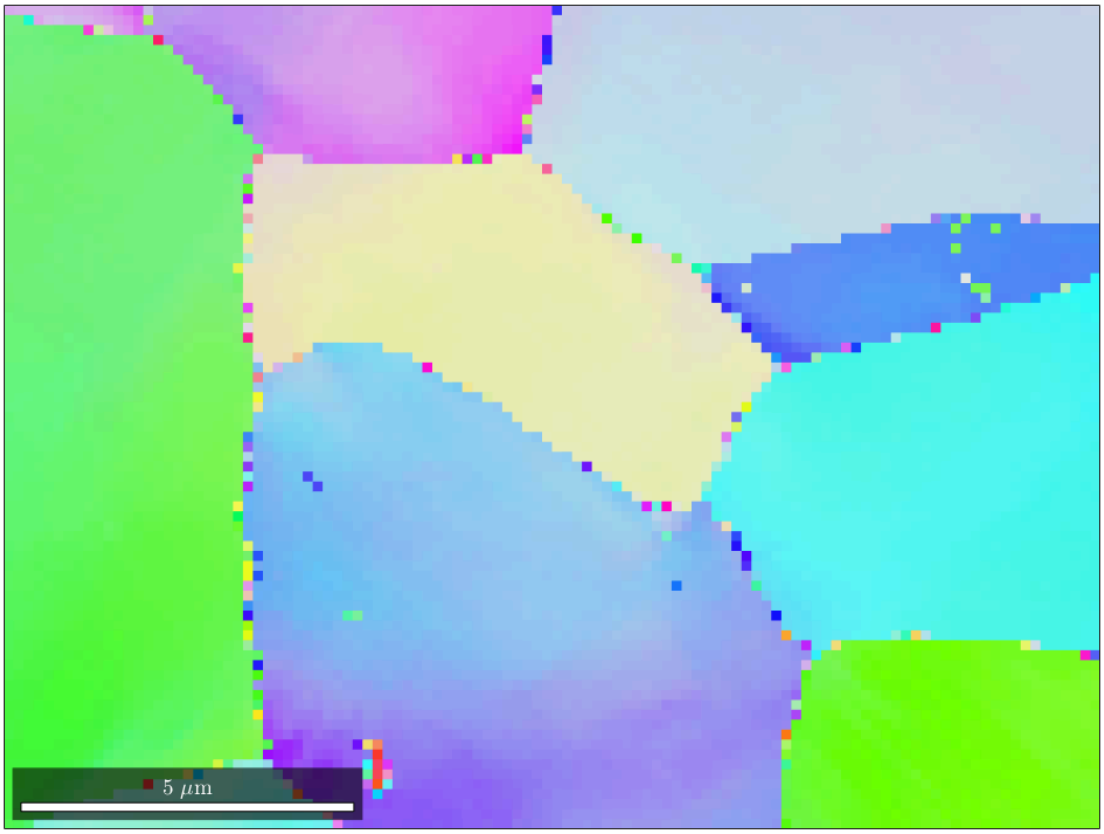
```
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'PCY',PCOut.Fit_2nd
 the data

%construct arrays from the indexed data
Astro_Phi1=zeros(MicroscopeData.NPoints,1);
Astro_PHI=zeros(MicroscopeData.NPoints,1);
Astro_Phi2=zeros(MicroscopeData.NPoints,1);
Astro_error=zeros(MicroscopeData.NPoints,1);
Astro_maxok=zeros(MicroscopeData.NPoints,1);

Astro_BC=Peak_Quality(:,1);
Astro_IQ=Peak_Quality(:,2);

for p=1:MicroscopeData.NPoints
    Astro_Phi1(p)=Indexed_Rotdata{p}.eang(1)*180/pi;
    Astro_Phi2(p)=Indexed_Rotdata{p}.eang(3)*180/pi;
    Astro_PHI(p)=Indexed_Rotdata{p}.eang(2)*180/pi;
    Astro_error(p)=Indexed_Rotdata{p}.error;
    Astro_maxok(p)=Indexed_Rotdata{p}.maxok;
end

%orientation data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'PHI',Astro_PHI); %
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'phi2',Astro_Phi2);
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'phi1',Astro_Phi1);
 the data

%beam data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'X
 BEAM',double(MapData.XBeam)-1); %subtract 1 from the data because of
 the 0 indexing
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'Y
 BEAM',double(MapData.YBeam)-1); %subtract 1 from the data because of
 the 0 indexing
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'X
 SAMPLE',MapData.XSample); %write the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'Y
 SAMPLE',MapData.YSample); %write the data

%indexing data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'MAD',Astro_error);
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'MADPhase',Astro_Ph
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'NIndexedBands',Ast
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,dtype,'RadonQuality',Astr
 the data

%header
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,htype,'CameraTilt',Micros
 the data
```

```
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,htype,'SampleTilt',Micros
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,htype,'TotalTilt',Microsc
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,htype,'KV',MicroscopeData
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,htype,'NCOLS',MicroscopeD
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,htype,'NROWS',MicroscopeD
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,htype,'XSTEP',MicroscopeD
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,htype,'YSTEP',MicroscopeD
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,htype,'NPoints',Microscop
 the data

h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,htype,'Magnification',Mic
 the data
h5_WritePair(OutputUser.HDF5FullFile,OutputUser.DataName,htype,'WD',MicroscopeData
 the data

Overwriting /Demo_Ben_16bin/EBSD/Data/DD
Overwriting /Demo_Ben_16bin/EBSD/Data/PCX
Overwriting /Demo_Ben_16bin/EBSD/Data/PCY
Overwriting /Demo_Ben_16bin/EBSD/Data/PHI
Overwriting /Demo_Ben_16bin/EBSD/Data/phi2
Overwriting /Demo_Ben_16bin/EBSD/Data/phi1
Overwriting /Demo_Ben_16bin/EBSD/Data/X BEAM
Overwriting /Demo_Ben_16bin/EBSD/Data/Y BEAM
Overwriting /Demo_Ben_16bin/EBSD/Data/X SAMPLE
Overwriting /Demo_Ben_16bin/EBSD/Data/Y SAMPLE
Overwriting /Demo_Ben_16bin/EBSD/Data/MAD
Overwriting /Demo_Ben_16bin/EBSD/Data/MADPhase
Overwriting /Demo_Ben_16bin/EBSD/Data/NIndexedBands
Overwriting /Demo_Ben_16bin/EBSD/Data/RadonQuality
Overwriting /Demo_Ben_16bin/EBSD/Header/CameraTilt
Overwriting /Demo_Ben_16bin/EBSD/Header/SampleTilt
Overwriting /Demo_Ben_16bin/EBSD/Header/TotalTilt
Overwriting /Demo_Ben_16bin/EBSD/Header/KV
Overwriting /Demo_Ben_16bin/EBSD/Header/NCOLS
Overwriting /Demo_Ben_16bin/EBSD/Header/NROWS
Overwriting /Demo_Ben_16bin/EBSD/Header/XSTEP
Overwriting /Demo_Ben_16bin/EBSD/Header/YSTEP
Overwriting /Demo_Ben_16bin/EBSD/Header/NPoints
Overwriting /Demo_Ben_16bin/EBSD/Header/Magnification
Overwriting /Demo_Ben_16bin/EBSD/Header/WD
```

Create stuff in MTEX

```
cs = loadCIF('Fe-Iron-alpha.cif'); %load the CIF file from MTEX cifs
setMTEXpref('xAxisDirection','west'); %set the axes conventions
setMTEXpref('zAxisDirection','outOfPlane'); %z is out of the page
```

```
% build the coordinate maps
prop.x = double(MapData.XSample);
prop.y = double(MapData.YSample);
ori = ...

 rotation('Euler',Astro_Phi1*degree,Astro_PHI*degree,Astro_Phi2*degree);

ebsd = EBSD(ori.', ones(size(ori)),{'notIndexed',cs},'options',prop);

figure;
colorKey = ipfHSVKey(cs);
colorKey.inversePoleFigureDirection = xvector;
color = colorKey.orientation2color(ebsd('indexed').orientations);

plot(ebsd,colorKey.orientation2color(ebsd.orientations))

% % If needed, plot an orientation color key,
% figure;
% plot(colorKey)

%end of script
```



Published with MATLAB® R2018b