Oving_1

October 8, 2024

1 Plotting av Focused Ion Beam data med matplotlib

Målet med denne Jupyter Notebooken er at dere skal lære hvordan man kan lage en figur med FIB data via `matplotlib`. Denne figuren skal inneholde minst 2 forskjellige bilder, hvor bare de relevante delene i bildene er med. Denne figuren skal også ha annoteringer, sånn som scalebar, a), b) og tekst som indikerer relevant objekter. Her så følger det med 3 eksempel bildefiler fra FIBen, denne Notebooken er designet for disse. Men det skal være enkelt for dere å tilpasse den til dataene dere har tatt opp. Tingene dere skal gjøre i denne øvingen er operasjoner som ganske lett kunne gjøres med vanlige bilderedigeringsprogrammer, men disse ferdighetene bygger et fundament for å jobbe med de mer avanserte dataene dere skal jobbe med i både TEM og SEM dataøvingene.

1.1 Visualisering

Først må vi velge visualiserings "backend", i denne øvingen skal vi bruke `%matplotlib inline`.

[ ]:
```
%matplotlib inline
```

2 Åpne data

La oss først kikke på en av TIFF filene fra FIBen. Disse kan vi f.eks. åpne ved hjelp av Python biblioteket `tifffile`. Importer dette biblioteket. Det er mange andre måter å åpne mikroskopi-data også, f.eks. ved å bruke biblioteket `HyperSpy`, som vi skal bruke neste dataøving som omhandler TEM.

Bruk funksjonen `imread` (som er i `tifffile` biblioteket) til å åpne `bilde0.tif` filen. Hvis du lurer på hvordan funksjonen virker, så kan du se på `docstring` til funksjonen ved å bruke `Shift` + `Tab`. Lagre denne til en variabel `bildedata0`. Her får vi opp noen verdiene som er i bildet. Dette er en `NumPy` array, som er et veldig viktig bibliotek i vitenskapelig Python.

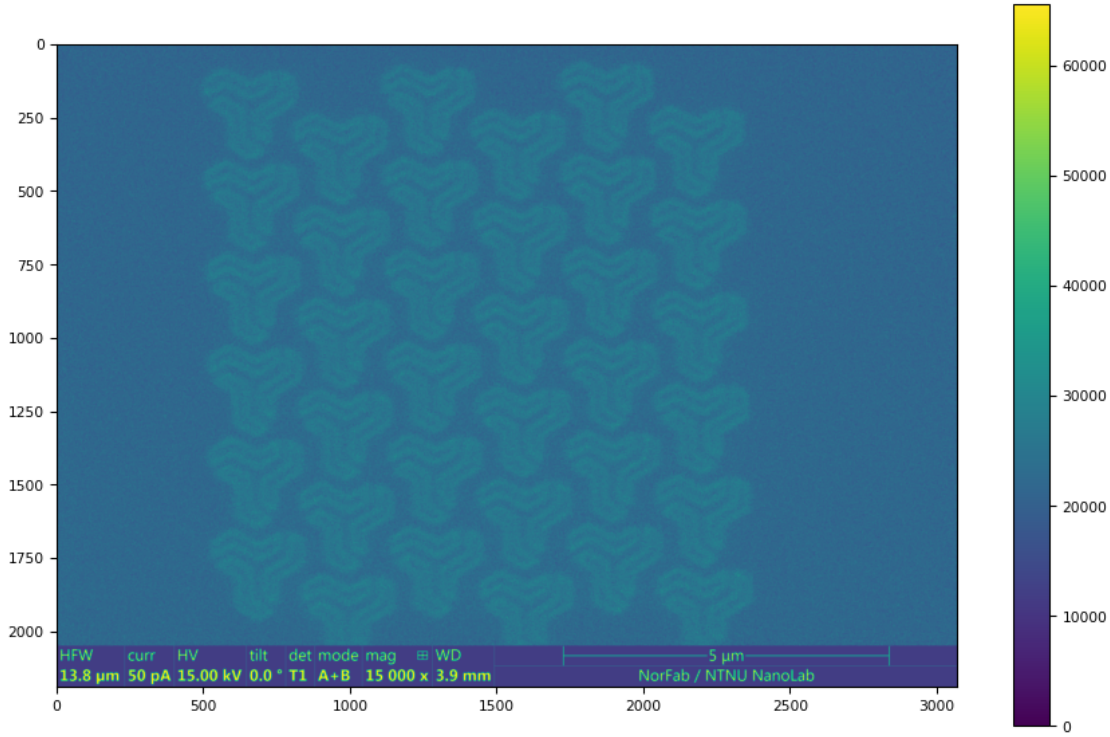
Men dette er ikke så altfor spennende! Sånne data er best å plotte i en figur.

`matplotlib` er et plotte-bibliotek i Python, som ofte brukes til å visualisere vitenskapelig data.

Importer sub-modulen `pyplot` i `matplotlib` og kall den `plt`. Så lag en figur av `bildedata0` ved å bruke `imshow` funksjonen i `plt`.

[ ]:
```
import tifffile as ti
bildedata0 = ti.imread('8c_far.tif')
import matplotlib.pyplot as plt
ti.imshow(bildedata0)
```

[ ]:
```
(<Figure size 988.8x604.8 with 2 Axes>,
 <Axes: >,
 <matplotlib.image.AxesImage at 0x1658236e0>)
```



2.1 Forbedre figuren: beskjæring

Først: beskjære bildet, slik at vi bare får med selve vinduet.

Vi gjør dette via NumPy slicing. Syntaksen her er `[y0:y1, x0:x1]` som virker på arrayen som skal beskjæres. `NUMPYARRAY[10:20, 10:20]`. Bruk bildet du allerede har lagd, til å finne `y0, y1, x0, x1`. Så lagre dette til en ny variabel: `bildedata0_c`.

Det kan være litt kronglete å finne de riktige verdiene for `y0, y1, x0, x1`. Noen tips:

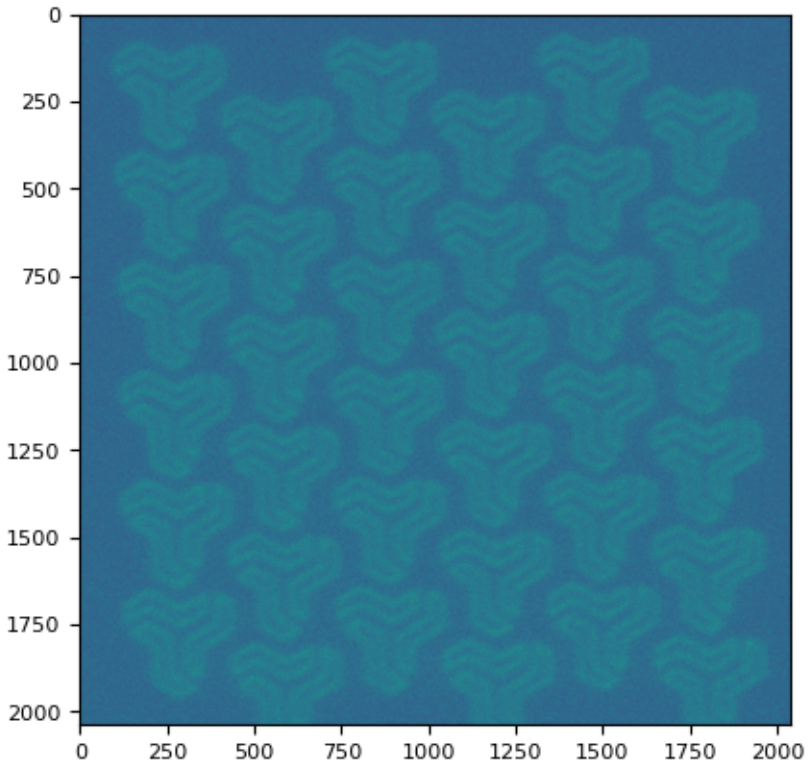
- Bruk tallene som står på aksene
- Bruk `bildedata0.shape` til å se hvor stort bildet er
- Prøv først slicingen med `NUMPYARRAY[:, :]`, og sett inn en av verdiene (`NUMPYARRAY[10:,:]`)
- Husk at du kan bruke negative indekser for å indeksere fra enden av NumPy arrays. F.eks. så vil `NUMPYARRAY[10:-10,:]` beskjære 10 fra starten og 10 fra enden.

[ ]:
```
bildedata0.shape
```

[ ]:
```
(2188, 3072)
```

```
[ ]: y0, y1 = 0, -150
     x0, x1 = 400, -634
     bildedata0_c = bildedata0[y0:y1, x0:x1,]
     #bildedata0_c = bildedata0_c[100:800, 100:1400] #fine cut
     plt.imshow(bildedata0_c)
     bildedata0_c.shape
```

```
[ ]: (2038, 2038)
```



For å lage fine figurer, så er det en fordel om bildene er like store (eller samme forhold mellom høyde og bredde). Når du er ferdig med beskjæringen, så hent ut den nye størrelsen, via `.shape` på den nye variablen (`bildedata0_c`).

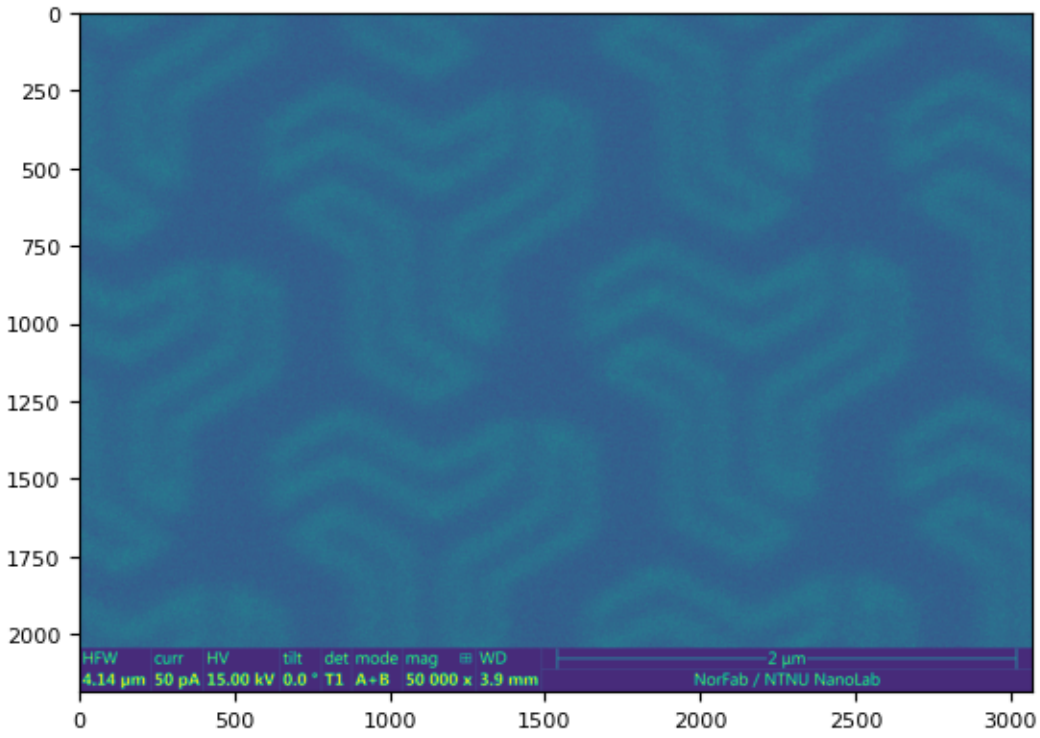
2.2 Bilde nummer 2

Nå kan vi legge til flere bilder i samme figur. Dette er veldig nyttig for effektiv bruk av plass i rapporter.

First åpne de en av de andre bildene (`bilde1.tif`), via `imread`. Kall den `bildedata1`. Deretter plot den via `plt.imshow`

```
[ ]: bildedata1 = ti.imread('8c_close.tif')
     plt.imshow(bildedata1)
```

[ ]: <matplotlib.image.AxesImage at 0x16523f3b0>



2.2.1 Rotere og beskjære

Her må vi roteter, slik at den blir horisontal.

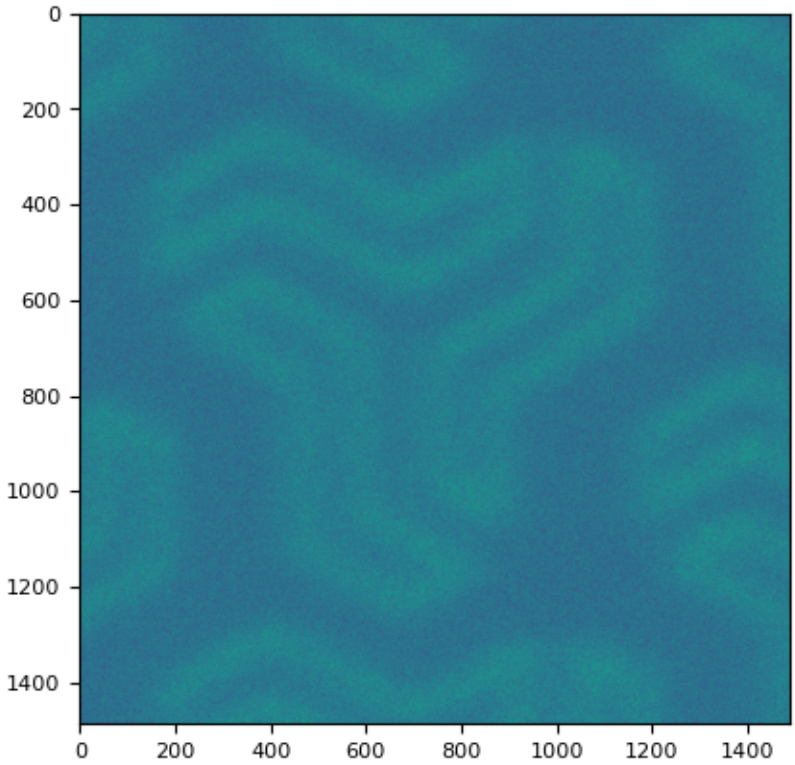
Først må vi importere en rotasjonsfunksjon fra `scipy`, denne er sub-modulen `ndimage`, og funksjonen heter `rotate`. Her kan `from ... import ...` brukes. Bruk `rotate`, til å gjøre vinduet horisontalt. Lag en ny variabel som heter `bildedata1_c`. Bruk docstring til å se hvordan den virker. Beskjær på samme måte som det forrige bildet, via numpy slicing.

[ ]:
```
from scipy.ndimage import rotate

bildedata1_c = rotate(bildedata1,angle=0)

y0, y1 = 0,-700
x0, x1 = 450,-1134
bildedata1_c = bildedata1_c[y0:y1, x0:x1,]
plt.imshow(bildedata1_c)
bildedata1_c.shape
```

[ ]: (1488, 1488)



2.3 Skalering av data

En viktig del av vitenskapelig data er metadata. Dette er informasjon om dataene: aksjelerasjonsspenning, detektortype, og romlig kalibrering. Det sistnevnte er spesielt viktig for plotting av figurene her, siden vi vil ha en kalibrert "scalebar" i bildet. Metadataene får vi tilgang til via `tifffile.TiffFile`, hvor selve filnavnet skal gis til `TiffFile`. Lag et `TiffFile` object `tif0`. Metadataene er i `fei_metadata` i `tif0`. Husk at du kan "minimere" resultater fra celler ved å trykke på den blå vertikale linjen til venstre for cellen. Merk: denne `tif0` er en dobbelt dictionary. Ergo, en dictionary som inneholder flere dictionary

[ ]:
```
tif0 = ti.TiffFile('8c_far.tif')
fei_metadata_0 = tif0.fei_metadata

for key, value in fei_metadata_0.items():
    print(f"{key}: {value}")
```

```
User: {'Date': '09/19/2024', 'Time': '01:30:40 PM', 'User': 'Supervisor',
'UserText': 'NorFab / NTNU NanoLab', 'UserTextUnicode': '4E006F00720046006100620
020002F0020004E0054004E00550020004E0061006E006F004C0061006200'}
System: {'Type': 'SEM', 'Dnumber': 9925407, 'Software': '13.9.0.6258',
'BuildNr': 6258, 'Source': 'FEG', 'Column': 'Nicole', 'FinalLens': 'Nicole',
'Chamber': 'xT-SSB', 'Stage': '110 x 110', 'Pump': 'TMP', 'ESEM': 'no',
'Aperture': 'AVA', 'Scan': 'PIA 3.0', 'Acq': 'PIA 3.0', 'EucWD': 0.01,
```

'SystemType': 'Apreo', 'DisplayWidth': 0.518, 'DisplayHeight': 0.324}
Beam: {'HV': 15000, 'Spot': 7, 'StigmatorX': -0.000861957, 'StigmatorY':
-0.000249983, 'BeamShiftX': -7.75085e-06, 'BeamShiftY': -1.54875e-05,
'ScanRotation': -2.59706, 'ImageMode': 'Normal', 'FineStageBias': 50, 'Beam':
'EBeam', 'Scan': 'EScan'}
EBeam: {'Source': 'FEG', 'ColumnType': 'Nicole', 'FinalLens': 'Nicole', 'Acq':
'PIA 3.0', 'Aperture': 'AVA', 'ApertureDiameter': 3.2e-05, 'HV': 15000, 'HFW':
1.38133e-05, 'VFW': 9.20889e-06, 'WD': 0.00394411, 'BeamCurrent': 5e-11,
'TiltCorrectionIsOn': 'no', 'DynamicFocusIsOn': 'no', 'DynamicWDIsOn': '',
'ScanRotation': -2.59706, 'LensMode': '', 'LensModeA': '', 'ATubeVoltage': 8000,
'UseCase': 'OptiPlan', 'SemOpticalMode': '', 'ImageMode': 'Normal',
'SourceTiltX': 0, 'SourceTiltY': 0, 'StageX': -0.000185357, 'StageY':
0.000773571, 'StageZ': 0.00400019, 'StageR': 0, 'StageTa': 1.40491e-05,
'StageTb': 0, 'StigmatorX': -0.000861957, 'StigmatorY': -0.000249983,
'BeamShiftX': -7.75085e-06, 'BeamShiftY': -1.54875e-05, 'EucWD': 0.01,
'EmissionCurrent': 0.000100806, 'TiltCorrectionAngle': 1.40491e-05, 'PreTilt':
0, 'WehneltBias': '', 'BeamMode': 'N-Beam', 'MagnificationCorrection': 'Off'}
GIS: {'Number': 0}
Scan: {'InternalScan': True, 'Dwelltime': 5e-06, 'PixelWidth': 4.49653e-09,
'PixelHeight': 4.49653e-09, 'HorFieldsize': 1.38133e-05, 'VerFieldsize':
9.20889e-06, 'Average': 0, 'Integrate': 1, 'FrameTime': 41}
EScan: {'Scan': 'PIA 3.0', 'InternalScan': True, 'Dwell': 5e-06, 'PixelWidth':
4.49653e-09, 'PixelHeight': 4.49653e-09, 'HorFieldsize': 1.38133e-05,
'VerFieldsize': 9.20889e-06, 'FrameTime': 41, 'LineTime': 0.02, 'Mainslock':
'On', 'LineIntegration': 1, 'ScanInterlacing': 1}
Stage: {'StageX': -0.00018675, 'StageY': 0.000790833, 'StageZ': 0.00400019,
'StageR': 0, 'StageT': 1.40491e-05, 'StageTb': 0, 'SpecTilt': 0,
'WorkingDistance': 0.00394411, 'ActiveStage': 'Bulk'}
Image: {'DigitalContrast': 1, 'DigitalBrightness': 0, 'DigitalGamma': 1,
'Average': 0, 'Integrate': 1, 'ResolutionX': 3072, 'ResolutionY': 2048,
'DriftCorrected': 'Off', 'ZoomFactor': 1.0, 'ZoomPanX': 0.5, 'ZoomPanY': 0.5,
'MagCanvasRealWidth': 0.2072, 'MagnificationMode': 1,
'ScreenMagCanvasRealWidth': 0.2072, 'ScreenMagnificationMode': 1,
'PostProcessing': 'None', 'Transformation': 'None'}
Vacuum: {'ChPressure': 0.000312621, 'Gas': '', 'UserMode': 'High vacuum',
'Humidity': ''}
Specimen: {'Temperature': '', 'SpecimenCurrent': -4.9992e-11}
Detectors: {'Number': 1, 'Name': 'T1', 'Mode': 'A+B'}
T1: {'Contrast': 32.6737, 'Brightness': 45.0979, 'Signal': 'BSE', 'ContrastDB':
39.2084, 'BrightnessDB': 45.0979, 'Setting': 'A+B', 'MinimumDwellTime': ''}
Accessories: {'Number': 0}
EBeamDeceleration: {'ModeOn': 'Off', 'LandingEnergy': 15000, 'ImmersionRatio':
1, 'StageBias': 0}
CompoundLensFilter: {'IsOn': 'Off', 'ThresholdEnergy': 0}
PrivateFei: {'BitShift': 0, 'DataBarSelected': 'mag curr HV tilt det mode WD
MicronBar Label', 'DataBarAvailable': 'srot frame dwell WD PW mag HFW x y z tilt
rotation pressure filter det DateTime mode driftCorr zoom spot UseCase curr HV
imRatio bias Aperture filterEnergy Label MicronBar', 'TimeOfCreation':

```
'19.09.2024 13:30:40', 'DatabarHeight': 0}
HiResIllumination: {'BrightFieldIsOn': '', 'BrightFieldValue': '',
'DarkFieldIsOn': '', 'DarkFieldValue': ''}
EasyLift: {'Rotation': 0}
HotStageMEMS: {'HeatingCurrent': '', 'HeatingVoltage': '', 'TargetTemperature':
'', 'ActualTemperature': '', 'HeatingPower': '', 'SampleBias': '',
'SampleResistance': ''}
HotStage: {'TargetTemperature': '', 'ActualTemperature': '', 'SampleBias': '',
'ShieldBias': ''}
HotStageHVHS: {'TargetTemperature': '', 'ActualTemperature': '', 'SampleBias':
'', 'ShieldBias': ''}
ColdStage: {'TargetTemperature': '', 'ActualTemperature': '', 'Humidity': '',
'SampleBias': ''}
```

Dette inneholder veldig mye informasjon, som er nyttig når man skal skrive om dataene i ettertid. Spesielt siden det er lett å glemme å skrive ned metadata.

Romlig kalibrering er i `['EScan']['PixelWidth']`, i dette tilfellet er `PixelWidth` og `PixelHeight` det samme. Så vi trenger bare en av dem. Lagre denne i en ny variabel: `skala0`. Gjenta dette for andre bilder

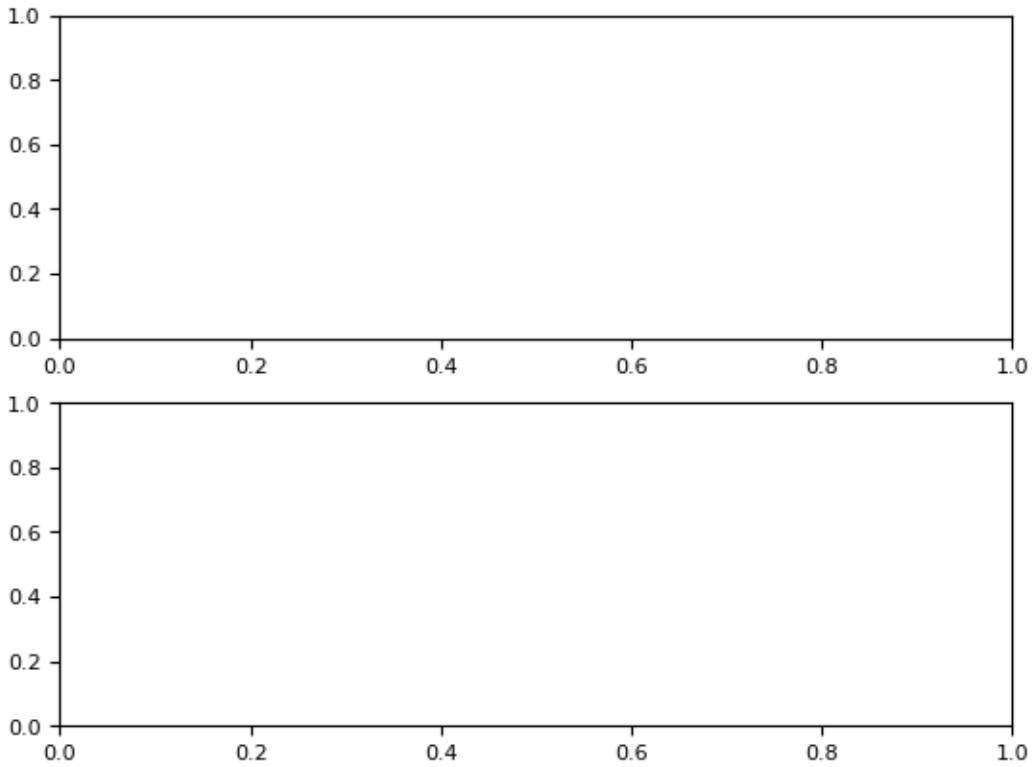
[ ]:
```
skala0 = fei_metadata_0['EScan']['PixelWidth']

tif1 = ti.TiffFile('8c_close.tif')
fei_metadata_1 = tif1.fei_metadata
skala1 = fei_metadata_1['EScan']['PixelWidth']
```

3 Lage plot med alle bildene

Nå som vi har gjort de 3 bildene klare, så kan vi kombinere dem i en figur. Første steget er å lage et figur objekt `fig` og en liste med 3 sub-plot objekter `axarr`. Dette lages via `plt.subplots` Tips: se på docstring med `Shift` + `Tab`, og bla ned til `Examples`. Her er vi ute etter å lage en figur, med tre vertikale subplot.

[ ]:
```
fig, axarr = plt.subplots(2,1)
```



Deretter "hent" ut de forskjellige subplottene fra `axarr`: `ax0`, `ax1`, `ax2`. Husk at denne er en liste.

[ ]:
```
ax0 = axarr[0]
ax1 = axarr[1]
```

Så kan vi bruke `imshow` på hvert av disse subplottene, til å plotte bildene. Men først må vi ordne skaleringen, siden vi vil plotte dataene i mikrometer. Dette gjøres via `extent` i `imshow`. Formattet er: `[left, right, bottom, top]`. Lag `extent0`, `extent1` og `extent2`, i form av en liste. Merk at dette er i meter, så det må endres til mikrometer.

[ ]:
```
extent0 = [0, skala0 * bildedata0_c.shape[1] * 10**6, 0, skala0 * bildedata0_c.
 ↪shape[0] * 10**6]
extent1 = [0, skala1 * bildedata1_c.shape[1] * 10**6, 0, skala1 * bildedata1_c.
 ↪shape[0] * 10**6]
```

Så kan vi bruke dette i `imshow`. Bruk `imshow` i `ax0`, `ax1` og `ax2`, kombinert med `extent=extent0`, `extent=extent1`, og `extent=extent2`. Tips: husk at det er veldig mye informasjon i docstrings! (`Shift` + `Tab`). For å se på resultatet, så vi må lagre dette som en bildefil. Dette gjøres via `savefig` i `fig` objektet. Figuren lagres via `savefig` i `fig` objektet: `bilde.jpg`

[ ]:
```
ax0.imshow(bildedata0_c,extent = extent0)
ax1.imshow(bildedata1_c,extent = extent1)
fig.savefig('bilde_1.jpg')
```

Åpne dette bildet, f.eks. ved å trykke på "Refresh File List" i JupyterLab, eller åpne filen direkte på datamaskinen.

Nå kan vi se hvordan figuren ser ut, og at det er en del ting som mangler.

- Oppløsningen er dårlig
- Masse ekstra "tomrom" som vi ikke bryr oss om
- Vi vil heller ha en "scale bar" enn å ha tallene på x- og y-aksen
- Det mangler annoteringer
- Bildene har litt forskjellige størrelser: dette kan løses ved å at beskjæringen tilpasses slik at alle har samme forhold mellom bredde og høyde.

3.1 Legge til en skalebar

Dette er litt komplisert kode, og bruker avanserte `matplotlib` funksjoner.

[ ]:
```
from mpl_toolkits.axes_grid1.anchored_artists import AnchoredSizeBar
import matplotlib.font_manager as fm
import matplotlib.patheffects as patheffects
fontprops = fm.FontProperties(size=12)
```

[ ]:
```
scalebar_kwargs = {'size': 2, 'label': '2 m', 'loc': 4, 'frameon': False,␣
 ↪'color': 'white', 'size_vertical': 0.05, 'label_top': False,␣
 ↪'fontproperties': fontprops}
scalebar0 = AnchoredSizeBar(transform=ax0.transData, **scalebar_kwargs)
# Denne legger til et svart omriss rundt scalebar teksten, for å gjøre den␣
 ↪lettere å lese
scalebar0.txt_label._text.set_path_effects([patheffects.withStroke(linewidth=2,␣
 ↪foreground='black', capstyle="round")])
ax0.add_artist(scalebar0)
```

[ ]:
```
<mpl_toolkits.axes_grid1.anchored_artists.AnchoredSizeBar at 0x168615e50>
```

[ ]:
```
scalebar_kwargs = {'size': 0.5, 'label': '500 nm', 'loc': 4, 'frameon': False,␣
 ↪'color': 'white', 'size_vertical': 0.01, 'label_top': False,␣
 ↪'fontproperties': fontprops}
scalebar1 = AnchoredSizeBar(transform=ax1.transData, **scalebar_kwargs)
# Denne legger til et svart omriss rundt scalebar teksten, for å gjøre den␣
 ↪lettere å lese
scalebar1.txt_label._text.set_path_effects([patheffects.withStroke(linewidth=2,␣
 ↪foreground='black', capstyle="round")])
ax1.add_artist(scalebar1)
```

[ ]:
```
<mpl_toolkits.axes_grid1.anchored_artists.AnchoredSizeBar at 0x167810aa0>
```

3.2 Legg til annoteringer

3.2.1 a, b og c

Legge til markering av subplottene, sånn som `a`, `b` og `c`. Dette kan gjøres via `ax` sin `annotate` funksjon. Sjekk docstring for informasjon. Bruk `xycoords='axes fraction'`, `fontsize` og `color`. Gjør dette for både `ax0`, `ax1` og `ax2`. Så lagre `bilde.jpg` på nytt.

[ ]:
```
ax0.annotate('A)',xy=(0.06,0.88),xycoords='axes␣
 ↪fraction',fontsize=12,color='white')
ax1.annotate('B)',xy=(0.06,0.88),xycoords='axes␣
 ↪fraction',fontsize=12,color='white')
```

[ ]:
```
Text(0.06, 0.88, 'B)')
```

3.2.2 Tall på objektene

Så legg til annoteringer på selve objektene, igjen ved å bruke `annotate`.

[ ]:
```
ax0.annotate('Pattern Sun',xy=(0.5,0.88),xycoords='axes␣
 ↪fraction',fontsize=10,color='white')
ax1.annotate('Single Sun',xy=(0.5,0.88),xycoords='axes␣
 ↪fraction',fontsize=10,color='white')
```

[ ]:
```
Text(0.5, 0.88, 'Single Sun')
```

3.3 Fjerne "akse-tall"

Fjerne tallene rundt plottet, ved å bruke `set_xticks([])` og `set_yticks([])` i `ax`-objektene.

[ ]:
```
ax0.set_xticks([]); ax0.set_yticks([])
ax1.set_xticks([]); ax1.set_yticks([])
```

[ ]:
```
[]
```

3.4 Fjerne tomrom i `fig`

Tilslutt, så fjerner vi tomrom rundt `fig` ved å bruke `subplots_adjust`. Det siste tomrommet kom fjernes ved å fin-innstille `figwidth` og `figheight` via `fig.set_figwidth` og `fig.set_figheight`. For å se hva den er nå, bruk `fig.get_figwidth()` og `fig.get_figheight()`

[ ]:
```
wid = fig.get_figwidth(); hig = fig.get_figheight()
print(f'Width: {wid}; Height: {hig}')
fig.subplots_adjust(hspace=0.06)
fig.savefig('bilde_2.jpg')
```

```
Width: 6.4; Height: 4.8
```

[ ]:
```
wid = fig.get_figwidth(); hig = fig.get_figheight()
print(f'Width: {wid}; Height: {hig}')
```

```
Width: 6.4; Height: 4.8
```

[ ]:
```
#fig.set_figwidth(1.5)
#fig.set_figheight(4.5)
fig.savefig('bilde_final.jpg',dpi=600,bbox_inches='tight')#pad_inches=0
```

4 Mer avansert plassering og design

Akkurat dette eksemplet var relativt enkelt, fordi de 3 sub-plottene hadde ganske lik størrelse og fasong. For andre typer data, så er det ikke tilfellet. For eksempel, hvis vi vil at bilde 0 og 1 skal være horisontalt, mens bilde 2 skal være vertikalt.

For dette kan vi bruke `subplot_mosaic`: https://matplotlib.org/stable/users/explain/axes/mosaic.html

- Matplotlib eksempler: https://matplotlib.org/stable/gallery/subplots_axes_and_figures/index.html
- Spesielt mosaic

Først skal må vi lage en `fig` variabel via `plt.figure`. Bruk `layout="constrained"`.

[ ]:

Deretter lag den følgende string:

```
mosaic = """
    AAC
    BBC
    """
```

[ ]:

Lag en ny variabel, `ax_dict` med å bruke `fig.subplot_mosaic` som tar `mosaic` string

[ ]:

Deretter hent ut `A`, `B` og `C` fra ax_dict, og lag de som variabel `ax_A`, `ax_B`, `ax_C`

[ ]:

Så bruk `imshow` til å plotte:

- `bildedata0_c` i `ax_A`
- `bildedata1_c` i `ax_B`

[ ]:

Så lag ett nytt bilde (`bildedata2_c_r`) ved å rotere `bildedata2_c`. Roter denne 90 grader ved å bruke `rotate` funksjonen fra tidligere.

[ ]:

[ ]:

Så lagre denne via `fig.savefig` som `mosaic_test.png`

5 Flere eksempler

Matplotlib har en egen "gallery" med eksempler for alt mulig rart. For mer avansert design og plasseringer av figurer, se https://matplotlib.org/stable/gallery/subplots_axes_and_figures/index.html

For alle eksemplene se https://matplotlib.org/stable/gallery/index.html

6 Deres egne data

Nå skal dere bruke denne Jupyter Notebooken til å lage tilsvarende figur, men med dataene dere tok opp i FIB-laben.

6.1 Måter figurene kan tilpasses:

- `cmap` parameteren i `imshow`. For eksempel: `ax.imshow(..., cmap='inferno')`. Se matplotlib sin dokumentasjon for en liste over fargekart. Merk at "perceptually uniform colormaps" er foretrukket i datavisualisering!
- Subplot posisjonering og størrelser kan kontrolleres med `add_subplot` og `GridSpec`
  - `add_subplot`: https://matplotlib.org/stable/tutorials/intermediate/gridspec.html#basic-quickstart-guide
  - `GridSpec`, mer komplisert, men mer kontroll: https://matplotlib.org/stable/tutorials/intermediate/grid adjustments-to-a-gridspec-layout

7 Eksportere som PDF

I JupyterLab: - File - Save and Export Notebook As - HTML - Åpne HTML filen i en nettleser - Via printing menyen i nettleseren: print til PDF

Lever denne PDFen + figurene i Blackboard

8 Fler eksempler på hva man kan gjøre med NumPy + matplotlib

- https://en.wikipedia.org/wiki/Quantum_dot#/media/File:Gaas_inas_quantum_dot.jpg
- https://en.wikipedia.org/wiki/Perovskite_(structure)#/media/File:Perovskite_oxide_thin_film.jpg med atomstruktur Atomic Simulation Environment (ASE)
- https://en.wikipedia.org/wiki/Scanning_transmission_electron_microscopy#/media/File:Scanning_transm
- https://en.wikipedia.org/wiki/Scanning_transmission_electron_microscopy#/media/File:Stem_dpc_schem
- https://en.wikipedia.org/wiki/Scanning_transmission_electron_microscopy#/media/File:Ferromagnetic_d
- https://en.wikipedia.org/wiki/Electron_energy_loss_spectroscopy#/media/File:Electron_energy_loss_sp
- https://en.wikipedia.org/wiki/Electron_energy_loss_spectroscopy#/media/File:Electron_energy_loss_sp