Import the required libraries

```python
%matplotlib widget
import hyperspy.api as hs
import numpy as np
import matplotlib.pyplot as plt
import atomap.api as am
hs.preferences.GUIs.warn_if_guis_are_missing = False
hs.preferences.save()
plt.rcParams['figure.figsize'] = (7,7)
```

Intensity Maps

First, we load the atom lattice:

```python
atom_lattice = am.load_atom_lattice_from_hdf5('atom_lattice_qw.hdf5')
s=hs.load('data.hspy')
sublattice_A=atom_lattice.sublattice_list[0]
sublattice_B=atom_lattice.sublattice_list[1]
```

Starting with the sublattice A, it is necessary to construct a 2D array with the intensities of every atom column in this sublattice. Therefore, we need to select the required planes from a zone axis.

Constructing the zone axes:

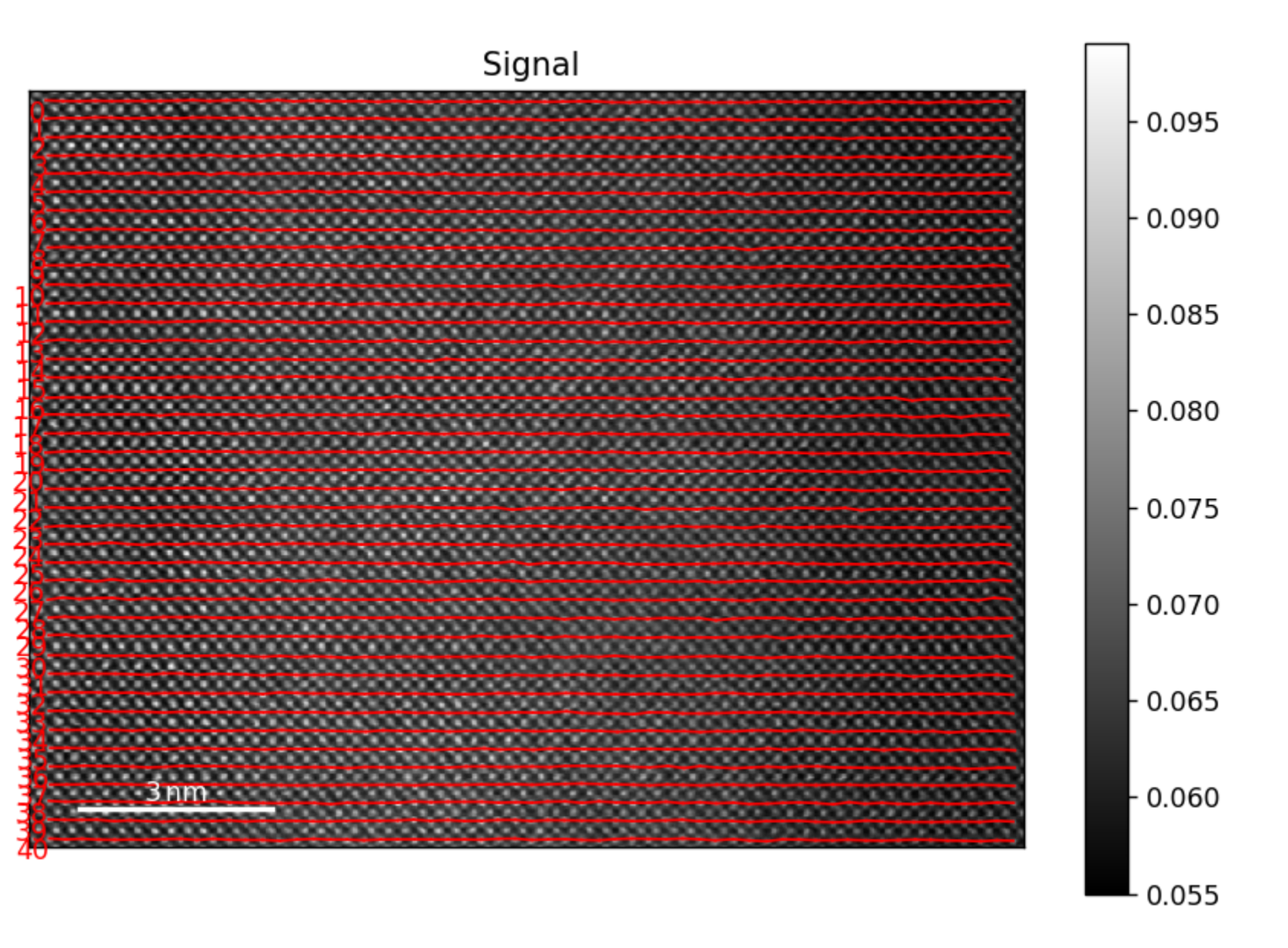
```python
sublattice_A.construct_zone_axes(atom_plane_tolerance=0.8)
```

To select the optimal zone axis, the parameter `direction` is varied until the optimum is found.

```python
direction=0

zone_vector = sublattice_A.zones_axis_average_distances[direction]
atom_planes = sublattice_A.atom_planes_by_zone_vector[zone_vector]
zone_axis = sublattice_A.get_atom_planes_on_image(atom_planes)
zone_axis.plot()
ax = plt.gca()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
sublattice_A
```

```
<Sublattice,  (atoms:2419,planes:6)>
```

Figure 1



Next, it is plotted the intensity of the atom columns in the sublattice A. We can select the required layers to plot with `first_layer` and `last_layer`.

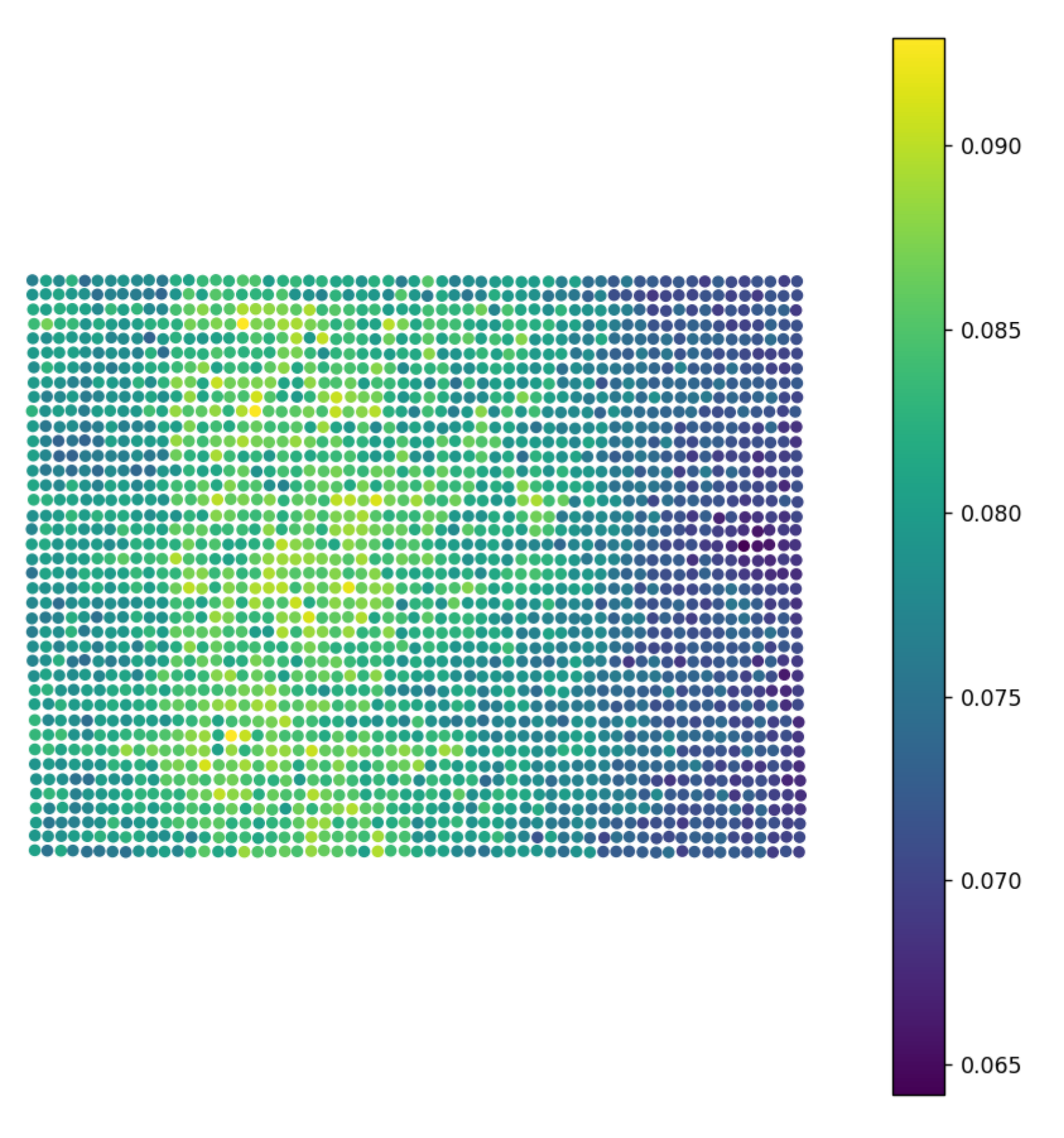
```python
first_layer = 0
last_layer = 39

sublattice_A.find_sublattice_intensity_from_masked_image(sublattice_A.image)
zone_axis_A = sublattice_A.zones_axis_average_distances[direction]
atom_plane_list_A = sublattice_A.atom_planes_by_zone_vector[zone_axis_A]
intensity_A=[]
x_values=[]
y_values=[]
for i in range(first_layer, last_layer+1):
    atomplane=atom_plane_list_A[i]
    plane_intensity=[]
    x_values_plane=[]
    y_values_plane=[]
    for j in range(0, len(atomplane.atom_list)):
        atom=atomplane.atom_list[j]
        x_pos,y_pos=atom.get_pixel_position()
        intensity=atom.intensity_mask
        plane_intensity.append(intensity)
        x_values_plane.append(x_pos)
        y_values_plane.append(y_pos)
    intensity_A.append(plane_intensity)
    x_values.append(x_values_plane)
    y_values.append(y_values_plane)

intensity_A_array = np.zeros([len(intensity_A),len(max(intensity_A,key = lambda x: len(x)))])
intensity_A_array[:] = np.nan
for i,j in enumerate(intensity_A):
    intensity_A_array[i][0:len(j)] = j

x_values_array = np.zeros([len(x_values),len(max(x_values,key = lambda x: len(x)))])
x_values_array[:] = np.nan
for i,j in enumerate(x_values):
    x_values_array[i][0:len(j)] = j

y_values_array = np.zeros([len(y_values),len(max(y_values,key = lambda x: len(x)))])
y_values_array[:] = np.nan
for i,j in enumerate(y_values):
    y_values_array[i][0:len(j)] = j

intensity_A=np.stack((intensity_A_array,x_values_array,y_values_array),axis=2)
np.save('intensity_A.npy',intensity_A)
```

The circles represent the atom columns of the sublattice A at their respective positions. But the circle sizes are not the same as the atom column sizes due to the splitting into two sublattices.

> Uncomment `plt.imshow(s.data,cmap='gray')` to additionally plot the original image.

```python
plt.figure()
plt.scatter(intensity_A[:,:,1],intensity_A[:,:,2],s=20,c=intensity_A[:,:,0])
plt.colorbar()
plt.axis('scaled')
plt.axis('off')
ax=plt.gca()
ax.set_ylim(ax.get_ylim()[::-1])
ax.xaxis.tick_top()
ax.yaxis.tick_left()
# plt.imshow(s.data,cmap='gray')
plt.tight_layout()
plt.show()
plt.savefig('intensitymap_sublatticeA.png',dpi=300,transparent=True,bbox_inches='tight')
```

Figure 2



For the sublattice B, the same process is followed.

Constructing the zone axes:

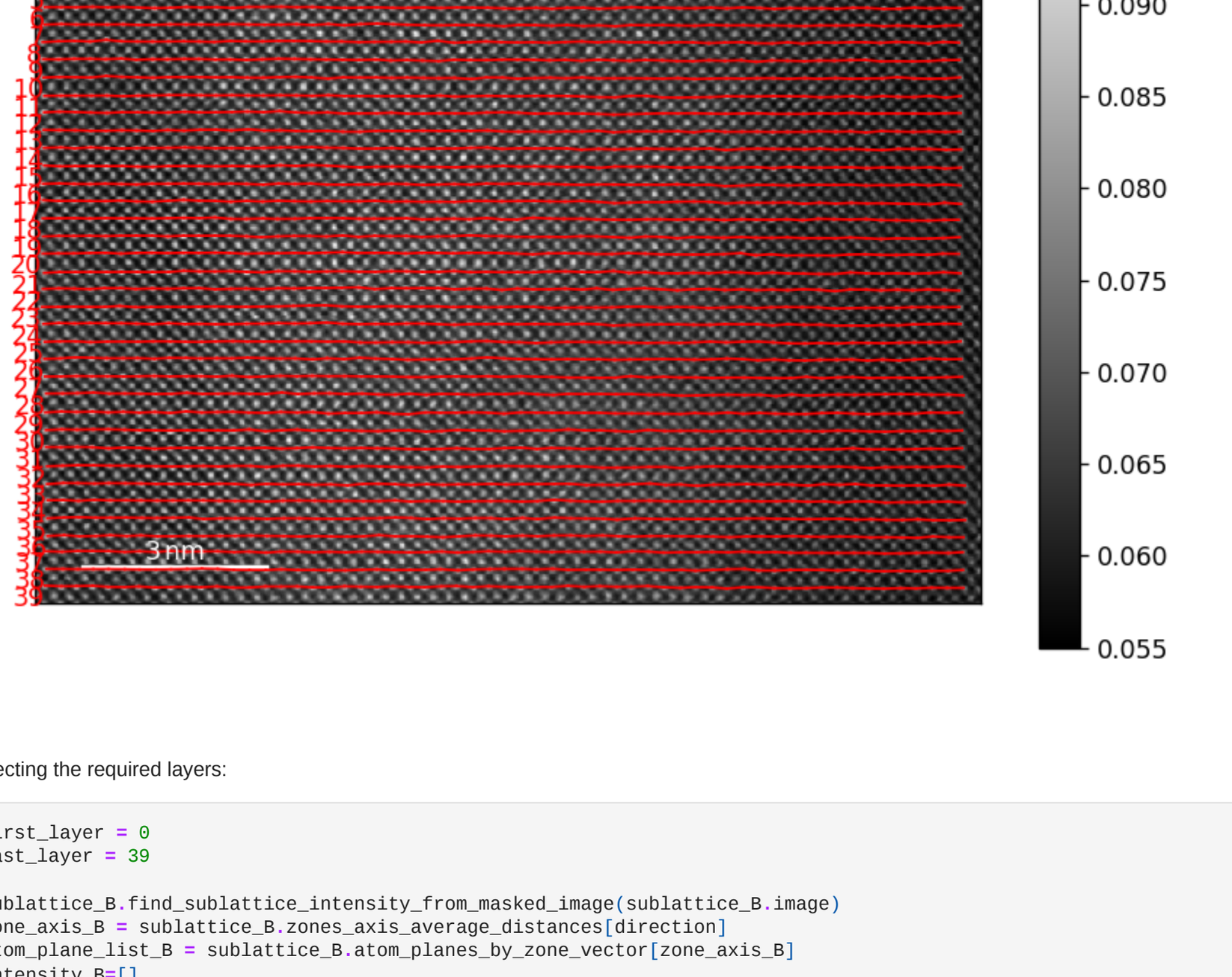
```python
sublattice_B.construct_zone_axes(atom_plane_tolerance=0.8)
```

Selecting the optimal zone axis:

```python
direction=0

zone_vector = sublattice_B.zones_axis_average_distances[direction]
atom_planes = sublattice_B.atom_planes_by_zone_vector[zone_vector]
zone_axis = sublattice_B.get_atom_planes_on_image(atom_planes)
zone_axis.plot()
ax = plt.gca()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
sublattice_B
```

```
<Sublattice,  (atoms:2360,planes:6)>
```

Figure 3



Selecting the required layers:

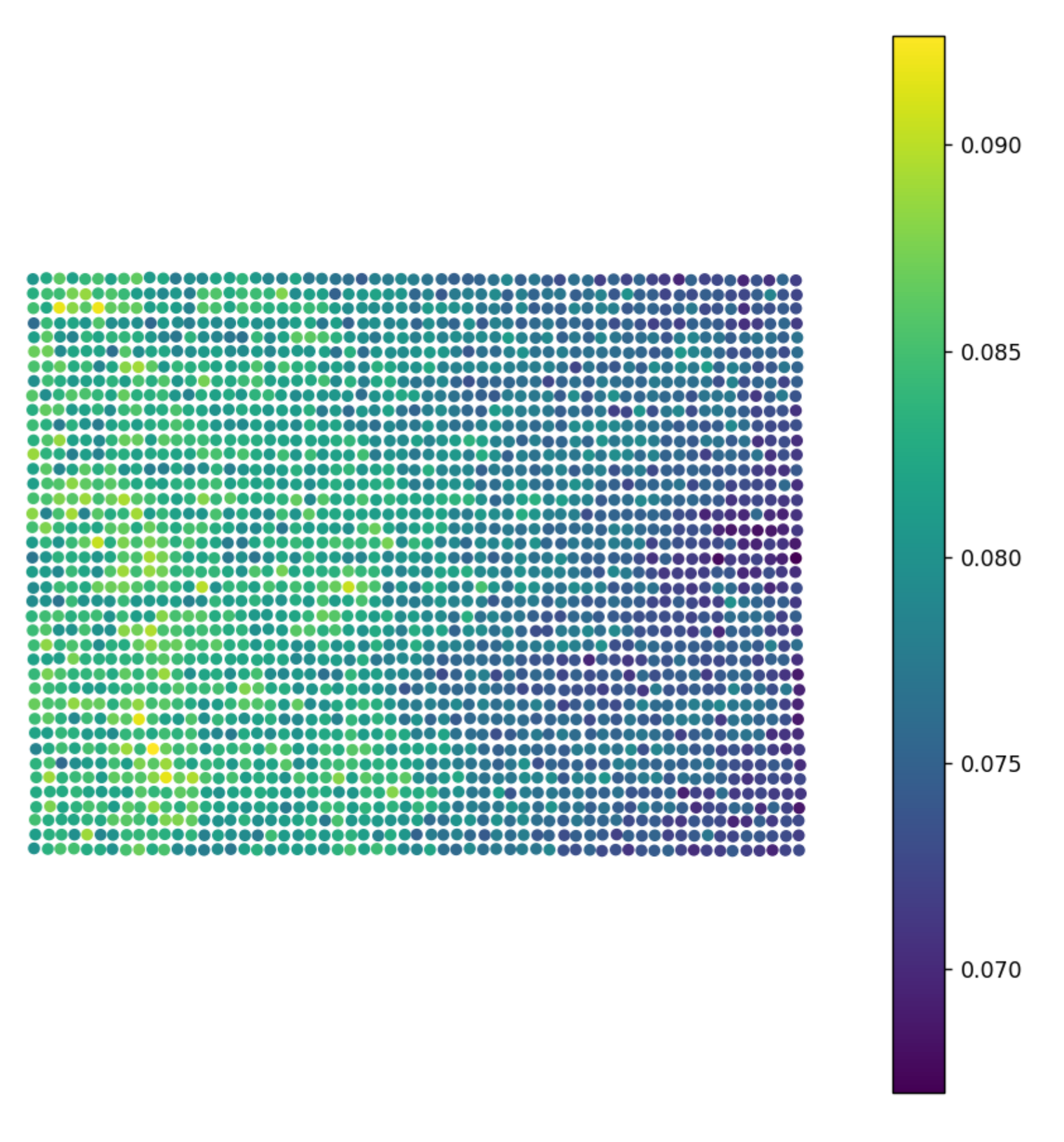
```python
first_layer = 0
last_layer = 39

sublattice_B.find_sublattice_intensity_from_masked_image(sublattice_B.image)
zone_axis_B = sublattice_B.zones_axis_average_distances[direction]
atom_plane_list_B = sublattice_B.atom_planes_by_zone_vector[zone_axis_B]
intensity_B=[]
x_values=[]
y_values=[]
for i in range(first_layer, last_layer+1):
    atomplane=atom_plane_list_B[i]
    plane_intensity=[]
    x_values_plane=[]
    y_values_plane=[]
    for j in range(0, len(atomplane.atom_list)):
        atom=atomplane.atom_list[j]
        x_pos,y_pos=atom.get_pixel_position()
        intensity=atom.intensity_mask
        plane_intensity.append(intensity)
        x_values_plane.append(x_pos)
        y_values_plane.append(y_pos)
    intensity_B.append(plane_intensity)
    x_values.append(x_values_plane)
    y_values.append(y_values_plane)

intensity_B_array = np.zeros([len(intensity_B),len(max(intensity_B,key = lambda x: len(x)))])
intensity_B_array[:] = np.nan
for i,j in enumerate(intensity_B):
    intensity_B_array[i][0:len(j)] = j

x_values_array = np.zeros([len(x_values),len(max(x_values,key = lambda x: len(x)))])
x_values_array[:] = np.nan
for i,j in enumerate(x_values):
    x_values_array[i][0:len(j)] = j

y_values_array = np.zeros([len(y_values),len(max(y_values,key = lambda x: len(x)))])
y_values_array[:] = np.nan
for i,j in enumerate(y_values):
    y_values_array[i][0:len(j)] = j

intensity_B=np.stack((intensity_B_array,x_values_array,y_values_array),axis=2)
np.save('intensity_B.npy',intensity_B)
```

Plotting the intensity map of the sublattice B:

> Uncomment `plt.imshow(s.data,cmap='gray')` to additionally plot the original image.

```python
plt.figure()
plt.scatter(intensity_B[:,:,1],intensity_B[:,:,2],s=20,c=intensity_B[:,:,0])
plt.colorbar()
plt.axis('scaled')
plt.axis('off')
ax=plt.gca()
ax.set_ylim(ax.get_ylim()[::-1])
ax.xaxis.tick_top()
ax.yaxis.tick_left()
# plt.imshow(s.data,cmap='gray')
plt.tight_layout()
plt.show()
plt.savefig('intensitymap_sublatticeB.png',dpi=300,transparent=True,bbox_inches='tight')
```

Figure 4



Thickness compensation

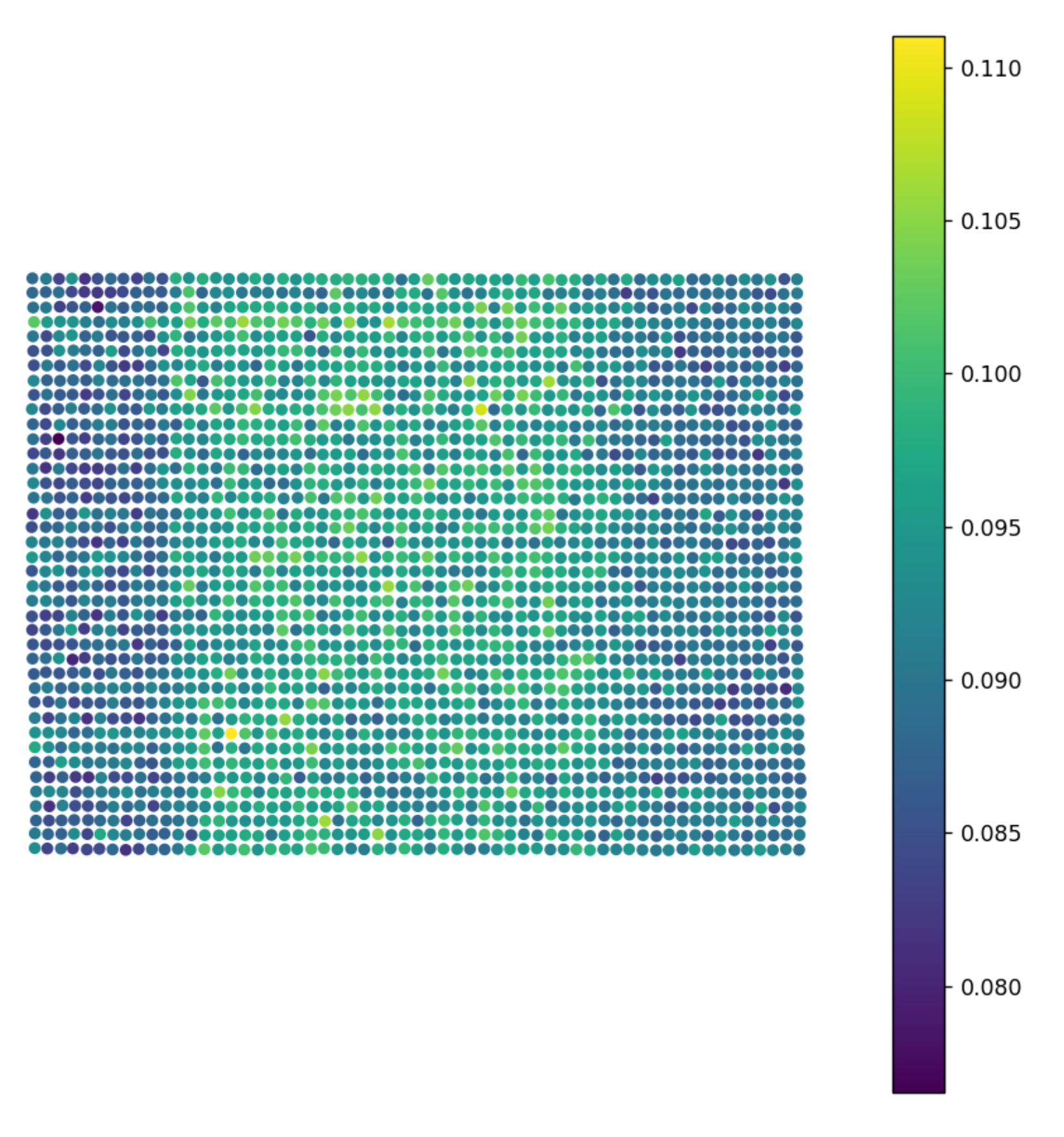
If the quantum well has only one element in a sublattice, we can use it to perform thickness compensation (for the other sublattice). It is mandatory that the two sublattices have the same dimensions because the 2D array of the group III sublattice will be divided element-by-element by the normalized 2D array of the group V sublattice.

$$I_{comp_{i,j}} = \frac{I_{gIII_{i,j}}}{I_{gVnorm_{i,j}}}$$

> `group_III_sublattice` and `group_V_sublattice` must be selected between `intensity_A` and `intensity_B`.
> Uncomment `plt.imshow(s.data,cmap='gray')` to additionally plot the original image.

```python
group_III_sublattice=np.copy(intensity_A)
group_V_sublattice=np.copy(intensity_B)

from atomap.atom_finding_refining import normalize_signal
normalized_V = normalize_signal(hs.signals.Signal2D(group_V_sublattice[:,:,0]))
compensated_III=np.divide(group_III_sublattice[:,:,0],normalized_V.data)
group_III_sublattice[:,:,0]=np.copy(compensated_III)
plt.figure()
plt.scatter(group_III_sublattice[:,:,1],group_III_sublattice[:,:,2],s=20,c=group_III_sublattice[:,:,0])
plt.colorbar()
plt.axis('scaled')
plt.axis('off')
ax=plt.gca()
ax.set_ylim(ax.get_ylim()[::-1])
ax.xaxis.tick_top()
ax.yaxis.tick_left()
# plt.imshow(s.data,cmap='gray')
plt.tight_layout()
plt.show()
plt.savefig('intensitymap_groupIII_compensated.png',dpi=300,transparent=True,bbox_inches='tight')
np.save('group_III_sublattice_comp.npy',group_III_sublattice)
np.save('group_V_sublattice.npy',group_V_sublattice)
```

Figure 5



Composition maps

The composition map of the segregated element present in the sublattice III is very similar to the intensity map, since to obtain it, the intensity map must be normalized according to the nominal composition of the element.

To obtain the composition map, the average intensities of two reference regions (the quantum well and the barriers) are calculated. Then, the intensity map is normalized with these two values and the nominal composition x_0 (Han et al., 2015):

$$x = \frac{x_0(I_{exp} - I_{Ga})}{I_{GaIn} - I_{Ga}}$$

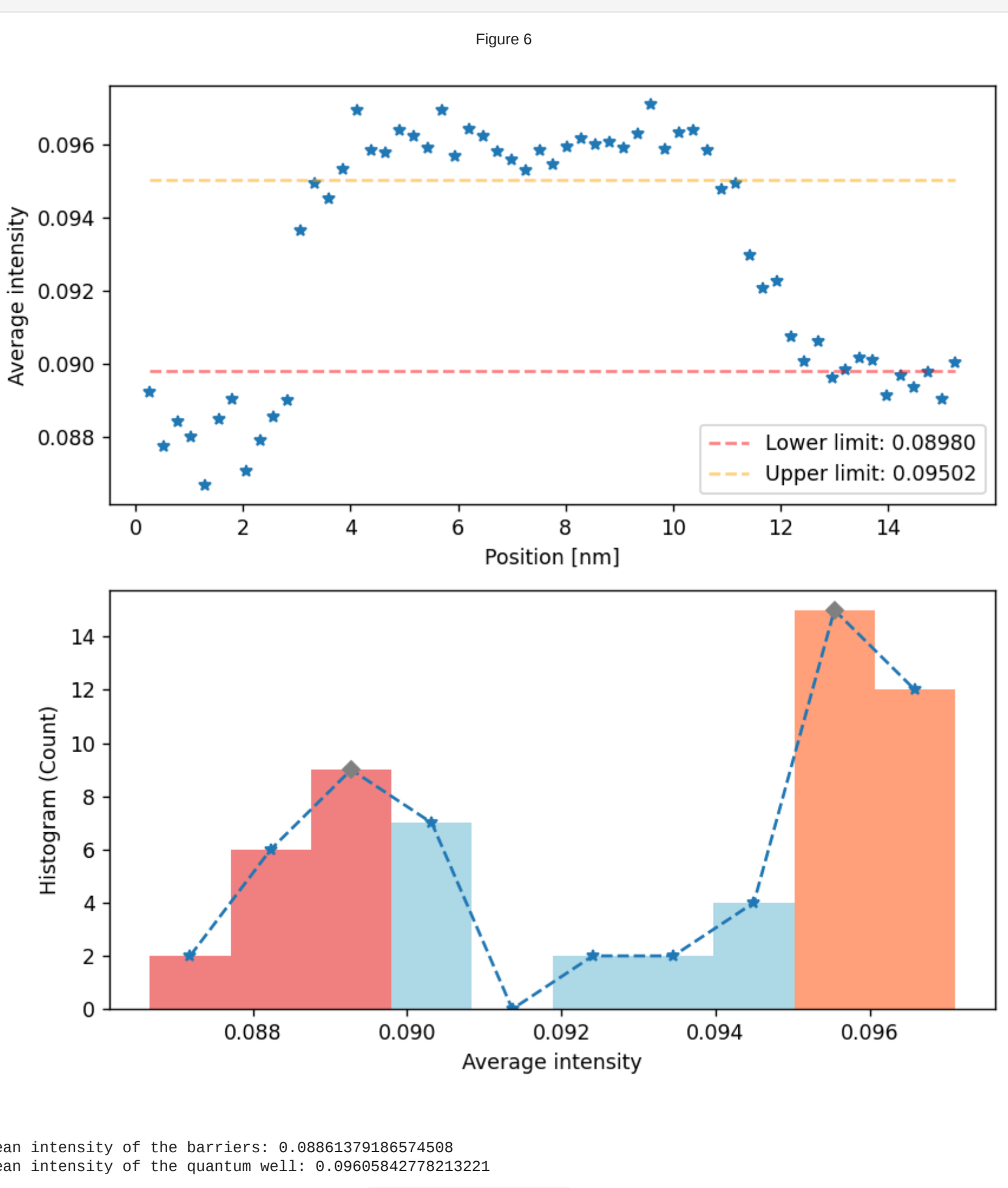
First, we select the `required_sublattice`:

```python
required_sublattice=np.copy(group_III_sublattice)
```

To get the average intensities, we select the upper and lower limit using the histogram of the average intensity plot:

```python
avg_i=np.mean(required_sublattice[:,:,0],axis=0)
avg_ax=np.mean(required_sublattice[:,:,1],axis=0)*s.axes_manager[0].scale

from scipy.stats import binned_statistic
from scipy.signal import find_peaks
count_binned=binned_statistic(avg_i,avg_i, 'count', bins=10)
bin_centers=(count_binned[1][1:] + count_binned[1][:-1])/2
mean_binned=binned_statistic(avg_i,avg_i, 'mean', bins=10)
pos_peaks, _ = find_peaks(count_binned[0], height=0)
pos_peaks_sorted=pos_peaks[np.argsort(count_binned[0][pos_peaks])]
peaks_sorted=mean_binned[0][pos_peaks_sorted]
max1,max2=np.sort(peaks_sorted[-2::])
max1_pos,max2_pos=np.where(mean_binned[0] == max1)[0],np.where(mean_binned[0] == max2)[0]
lower_limit=count_binned[1][max1_pos+1]
upper_limit=count_binned[1][max2_pos]

fig, (ax1, ax2) = plt.subplots(2, 1)
ax1.plot(avg_ax,avg_i,'*')
ax1.set(xlabel='Position ['+s.axes_manager[0].units+']',ylabel='Average intensity')
label1='Lower limit: %.5f' % (lower_limit)
label2='Upper limit: %.5f' % (upper_limit)
ax1.plot(avg_ax,[lower_limit]*len(avg_i),linestyle='--',color='red',alpha=0.5,label=label1)
ax1.plot(avg_ax,[upper_limit]*len(avg_i),linestyle='--',color='orange',alpha=0.5,label=label2)
ax1.legend()
_, _, patches = ax2.hist(avg_i,bins=count_binned[1],color='lightblue')
for i in (np.where(np.arange(0,len(patches))<=max1_pos))[0]:
    patches[int(i)].set_fc('lightcoral')
for i in (np.where(np.arange(0,len(patches))>=max2_pos))[0]:
    patches[int(i)].set_fc('lightsalmon')
ax2.plot(bin_centers,count_binned[0],'*--')
ax2.plot(bin_centers[pos_peaks], count_binned[0][pos_peaks], 'D',color='gray')
ax2.set(xlabel='Average intensity',ylabel='Histogram (Count)')
fig.tight_layout()
plt.show()

positions_l=np.where(avg_i<lower_limit)
i_barriers=np.nanmean(avg_i[positions_l])
positions_u=np.where(avg_i>upper_limit)
i_quantum_well=np.nanmean(avg_i[positions_u])
print('Mean intensity of the barriers: '+str(i_barriers))
print('Mean intensity of the quantum well: '+str(i_quantum_well))
```

Figure 6

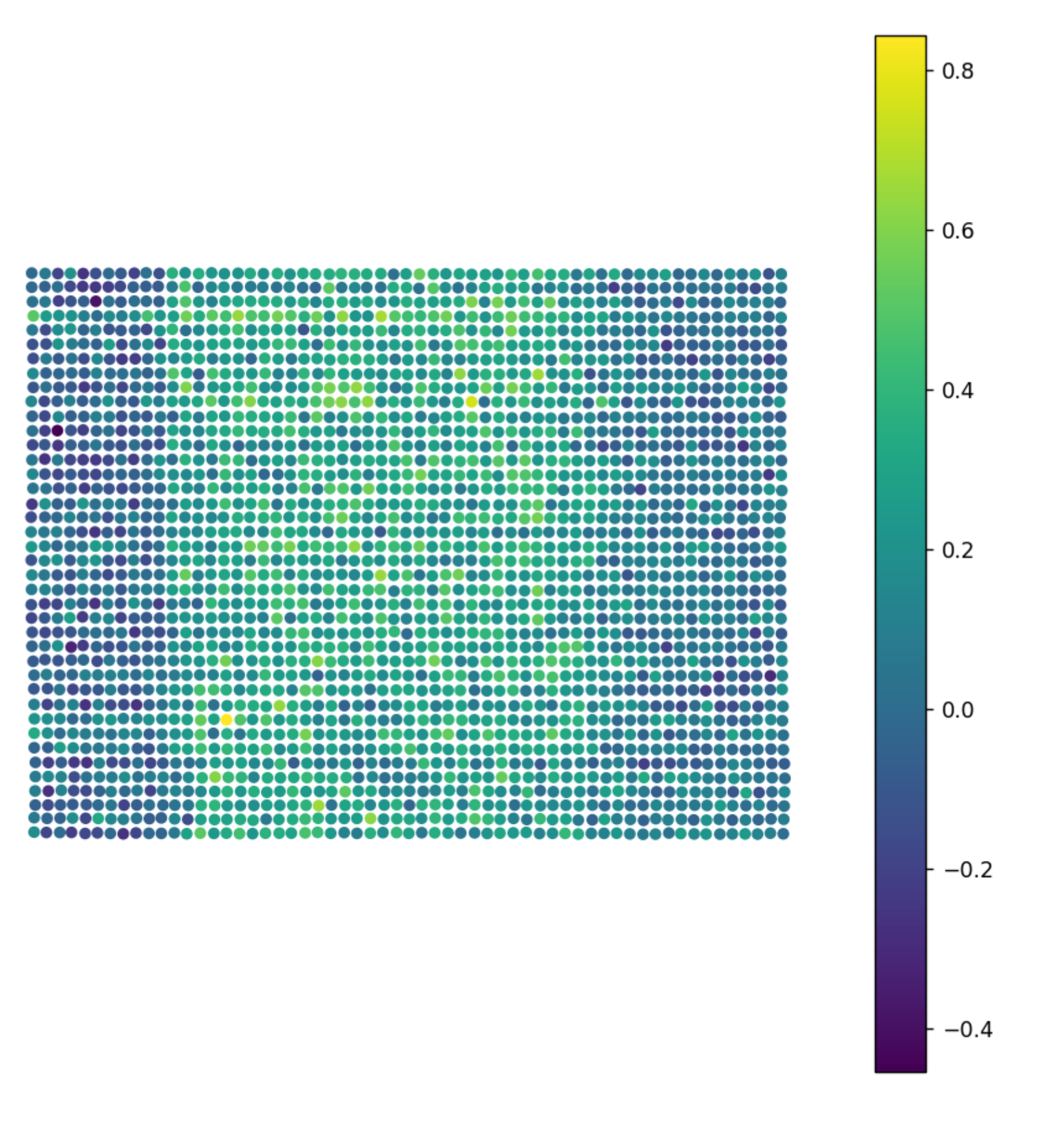



```
Mean intensity of the barriers: 0.08861379186574508
Mean intensity of the quantum well: 0.09605842778213221
```

Finally, we can obtain the composition map using the `nominal_composition`:

```python
nominal_composition=0.28

composition_map=np.copy(required_sublattice)
composition_map[:,:,0]=nominal_composition*(required_sublattice[:,:,0]-i_barriers)/(i_quantum_well-i_barriers)
plt.figure()
plt.scatter(composition_map[:,:,1],composition_map[:,:,2],s=20,c=composition_map[:,:,0])
plt.colorbar()
plt.axis('scaled')
plt.axis('off')
ax=plt.gca()
ax.set_ylim(ax.get_ylim()[::-1])
ax.xaxis.tick_top()
ax.yaxis.tick_left()
#plt.imshow(s.data,cmap='gray')
plt.tight_layout()
plt.show()
plt.savefig('composition_map.png',dpi=300,transparent=True,bbox_inches='tight')
np.save('composition_map.npy',composition_map)
```

Figure 7

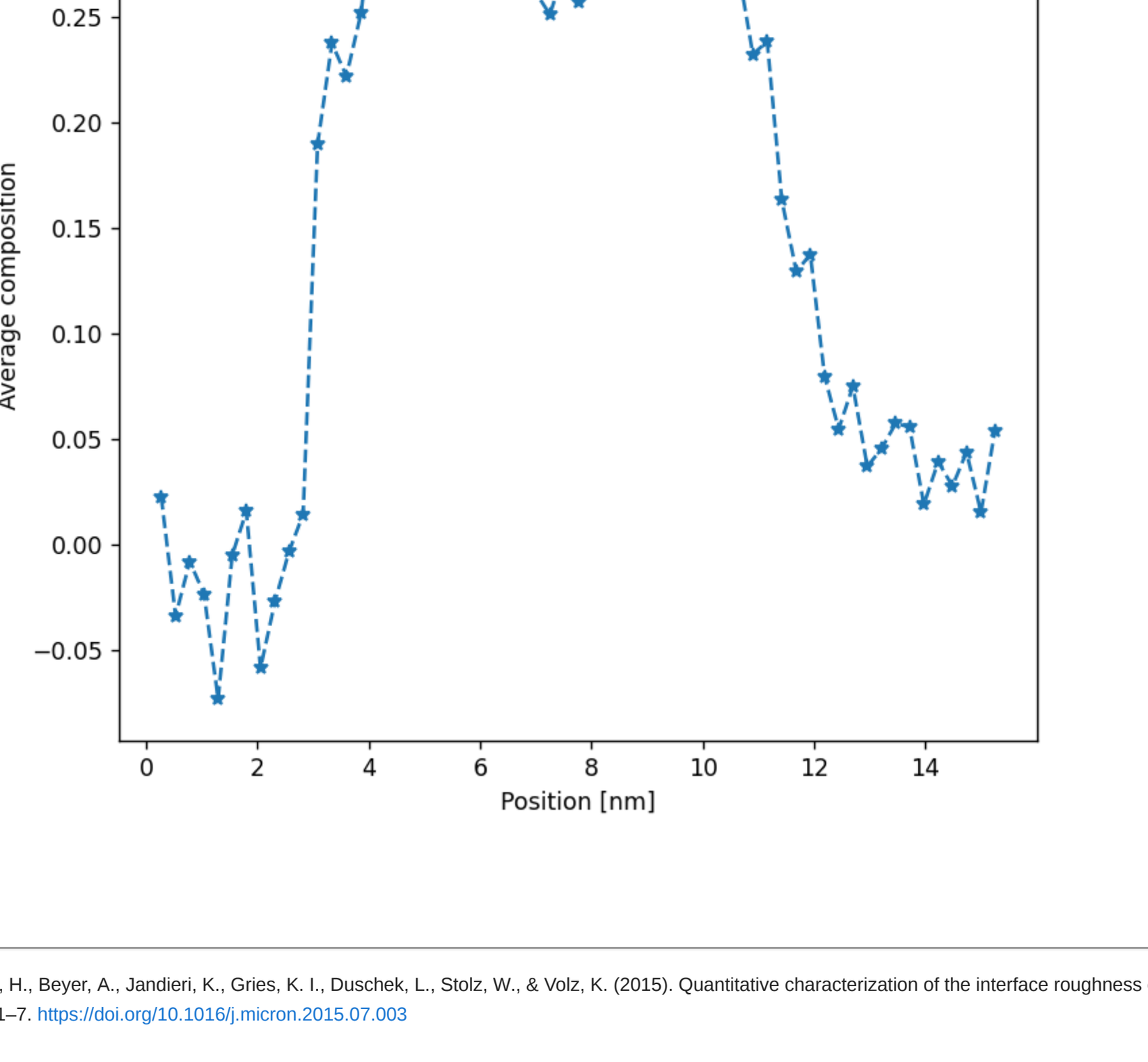


Composition profile

In addition to the composition map, we can average the perpendicular layers to the growth direction in order to get the composition profile:

```python
avg_intensity=np.mean(composition_map[:,:,0],axis=0)
avg_axis=np.mean(composition_map[:,:,1],axis=0)*s.axes_manager[0].scale
plt.figure()
plt.plot(avg_axis,avg_intensity,'*',linestyle='--')
plt.xlabel('Position ['+s.axes_manager[0].units+']')
plt.ylabel('Average composition')
plt.show()
```

Figure 8



Han, H., Beyer, A., Jandieri, K., Gries, K. I., Duschek, L., Stolz, W., & Volz, K. (2015). Quantitative characterization of the interface roughness of (GaIn)As quantum wells by high resolution stem. Micron, 79, 1–7. https://doi.org/10.1016/j.micron.2015.07.003