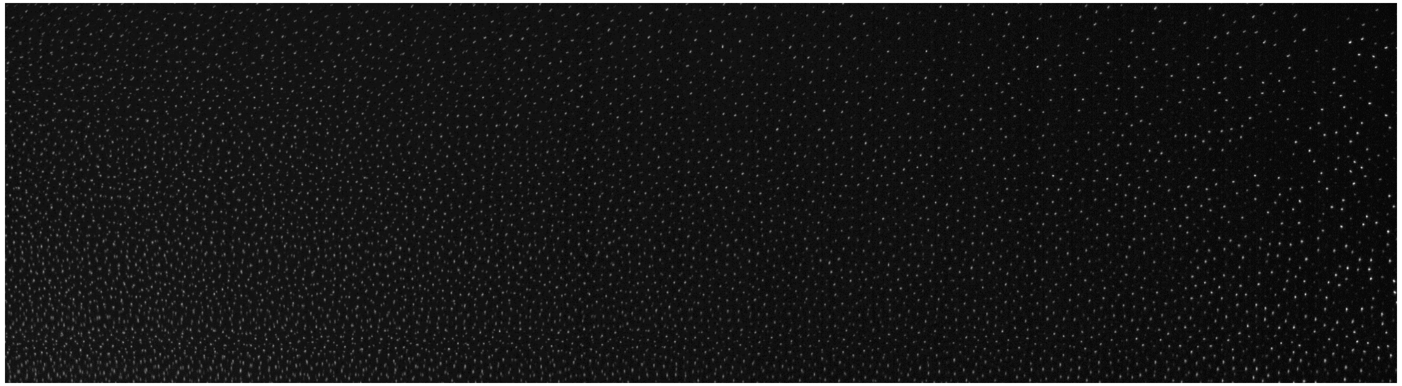
Particle Number Density Determination

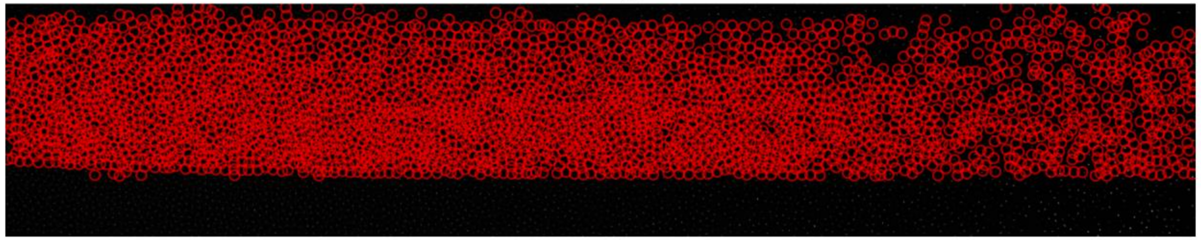
Utilized image reconstruction to determine the particle number density n_d.

Crystalline Structure Formation:

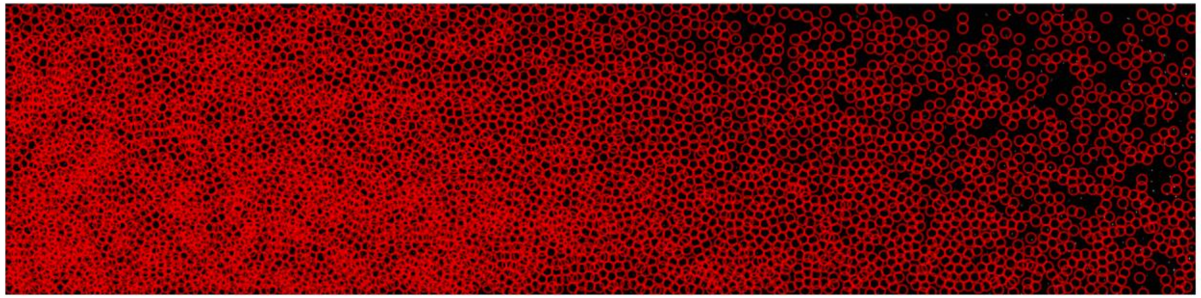
- Employed a stopping-process via polarity switching or RF-coil
- Resulted in the formation of a crystalline structure composed of particles.



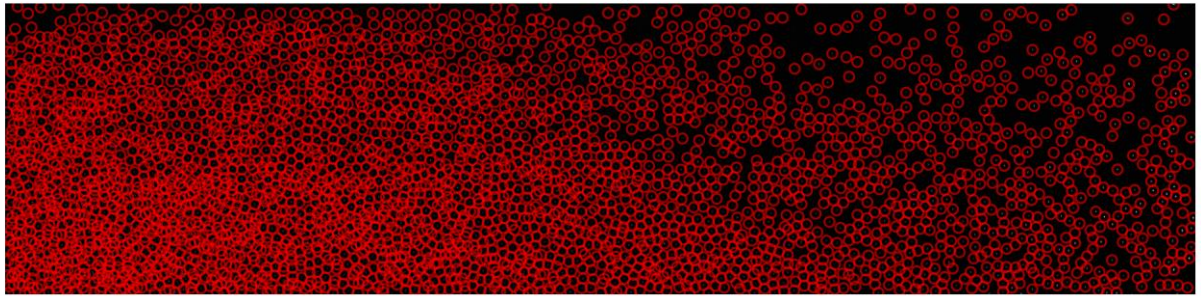
Particle Tracking 30 Pascal



Particle Tracking 40 Pascal



Particle Tracking 50 Pascal



Density Determination Methodology:

- Extracting coordinates via tracking code. Careful parameter selection is vital to avoid errors and exclude faint particles at different planes, see bottom image.

```
### 1.31 micrometer particles do have a psf of around 3 pixels [M.Y. Pustylnik et. al. (2016)]
ps_in_px = 9   #point spread function of particle; needs to be odd number (13 from agglomeration 24.06.2022, 9 from density 16.02.2023)
min_mass_cut = 70   #Brigthness Threshold = cancel out faint particles; from experience (200 from agglomeration 24.06.2022, 70 from density 16.02.2023)
iter_step_dr = 0.5  #from experience (accuracy and computationsla time taken into account)
```

Density Determination Methodology:

- Utilized data from particle tracking to determine the pair correlation function with density_2D_vsf [https://link.aps.org/doi/10.1103/PhysRevResearch.2.033404].

- Assumed a Wigner-Seitz cell to assess three-dimensional density [A. Melzer, Physics of Dusty Plasmas]..

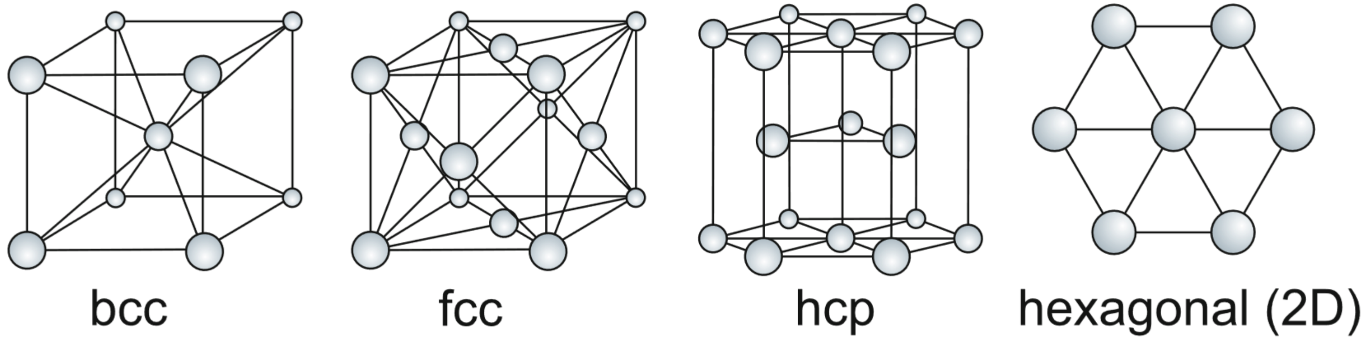
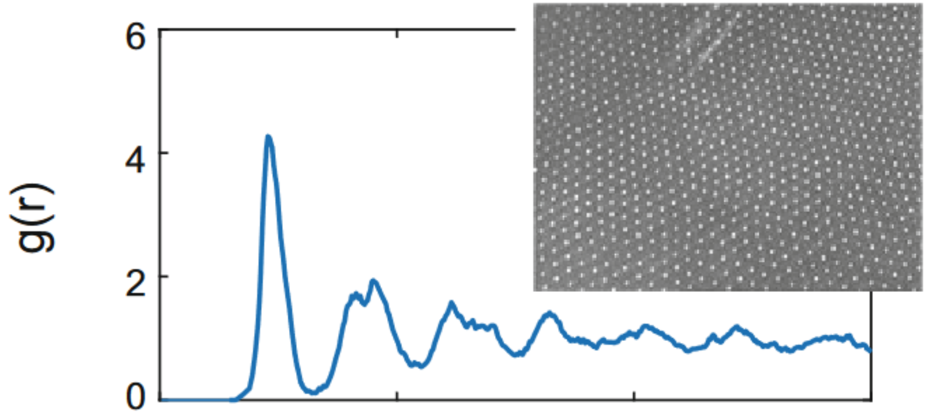


Fig. 5.1 Crystal structures in 3D: BCC, FCC and HCP. In 2D: hexagonal crystal structure. The BCC structure contains particles at the edges of a cube and an additional particle in the cube center. Likewise FCC contains additional particles at the cube faces. HCP is very similar to FCC where the ordering repeats every other layer instead of every third layer as in FCC



Mean Radius and Wigner-Seitz Radius Calculation:

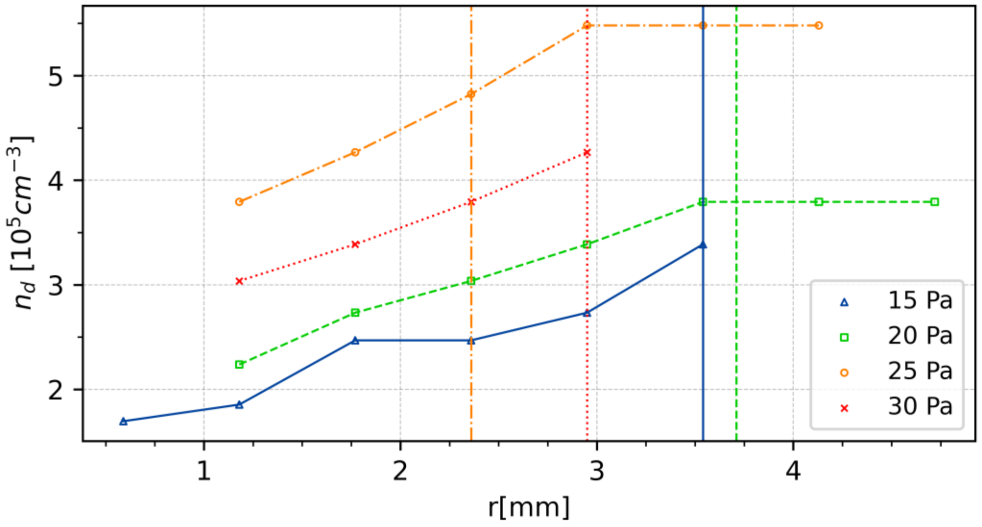
- The first peak in the distribution corresponds to the mean radius r_0.
- Calculated the Wigner-Seitz radius a=r0/ρ with ρ=1.79±0.07 from simulations.

Particle Number Density Computation:

- Computed particle number density n_d=3(4πa^3).
- Associated with a maximum uncertainty of up to ±12% [https://doi.org/10.1063/1.4914468].

Visualization:

- Figure illustrates the radial distribution of n_d from the tube axis.
- Measurements conducted at various pressures.



More Code Insight:

```
### PREPERATION ###

#Frame pre analysis. Prepare frame for density calculations. Crop Image at particle population area.
frames = pims.open('densitydata/*.bmp')

img1 = frames[0]

cut_l = 1250
cut_r = 1650

img_1 = img1[cut_l:cut_r,:]


# Optionally, tweak styles.
mpl.rc('figure',  figsize=(10, 5))
mpl.rc('image', cmap='gray')
plt.imshow(img_1)

#%%

img_1 = gaussian_filter(img_1, sigma=1)
plt.imshow(img_1)
```

Read in and cut out the interesting part of the image. Find the correct cuts individually.

Apply gaussian_filter to improve quality and reduce noise.

More Code Insight:

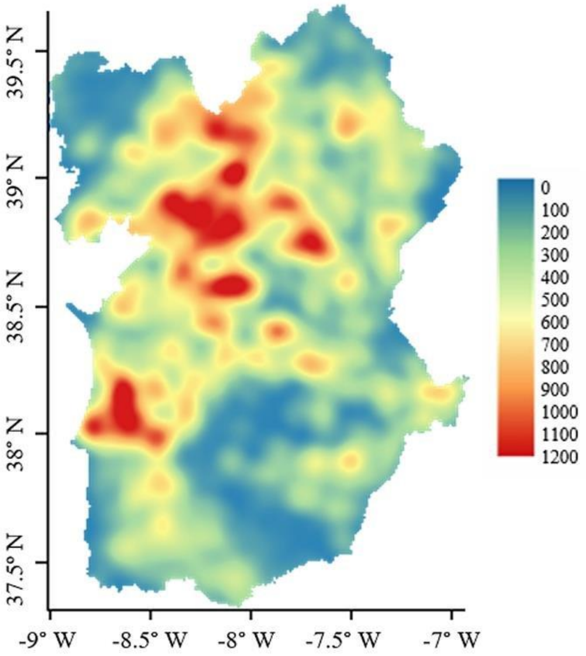
```
#https://michael-fuchs-python.netlify.app/2020/06/20/hdbscan/

#PSF = 3-12 depending on particle size and exposure time
#Brigthness Threshold = cancel out faint particles, depending on camera gain and exposure time
#doi = {10.1063/1.4914468}

df_located = tp.locate(img_1,9,70)      #(img, PSD, min_brigthness)

#
x_coords = df_located['x'].to_numpy()
y_coords = df_located['y'].to_numpy()
#
data = np.transpose(np.vstack((x_coords,y_coords)))

#PLOT
plot_kwds = {'alpha' : 0.25, 's' : 25, 'linewidths':1}
fig,ax = plt.subplots(dpi=600)
plt.imshow(img_1)
plt.scatter(data.T[0], data.T[1], color='r', **plot_kwds, facecolors='none')
```

Preadjust PSD and brightness threshold precisely

-> Go to ###Main Code### and make it run -> Last step, create the plot.

Future Add On:

– Visualizing by 2D density mapping



Here is my aim at a more complete answer including choosing the color map and a logarithmic normalization of the color axis.

```
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from matplotlib.colors import LogNorm
import numpy as np
x, y, z = np.loadtxt('data.txt', unpack=True)
N = int(len(z)**.5)
z = z.reshape(N, N)
plt.imshow(z+10, extent=(np.amin(x), np.amax(x), np.amin(y), np.amax(y)),
        cmap=cm.hot, norm=LogNorm())
plt.colorbar()
plt.show()
```

I assume here that your data can be transformed into a 2d array by a simple reshape. If this is not the case than you need to work a bit harder on getting the data in this form. Using imshow and not pcolormesh is more efficient here if you data lies on a grid (as it seems to do). The above code snippet results in the following image, that comes pretty close to what you wanted: