In [1]:
```
#Exploring distance of mitochondria to chloroplasts and Cell Walls
```

In [2]:
```
from sfepy.discrete.fem import Mesh, FEDomain, Field
import os
from stl import mesh
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import pyvista as pv
import numpy as np
import networkx as nx
from scipy.spatial import KDTree
import seaborn as sns
import vtk
import time
from IPython.display import display
import itkwidgets
```

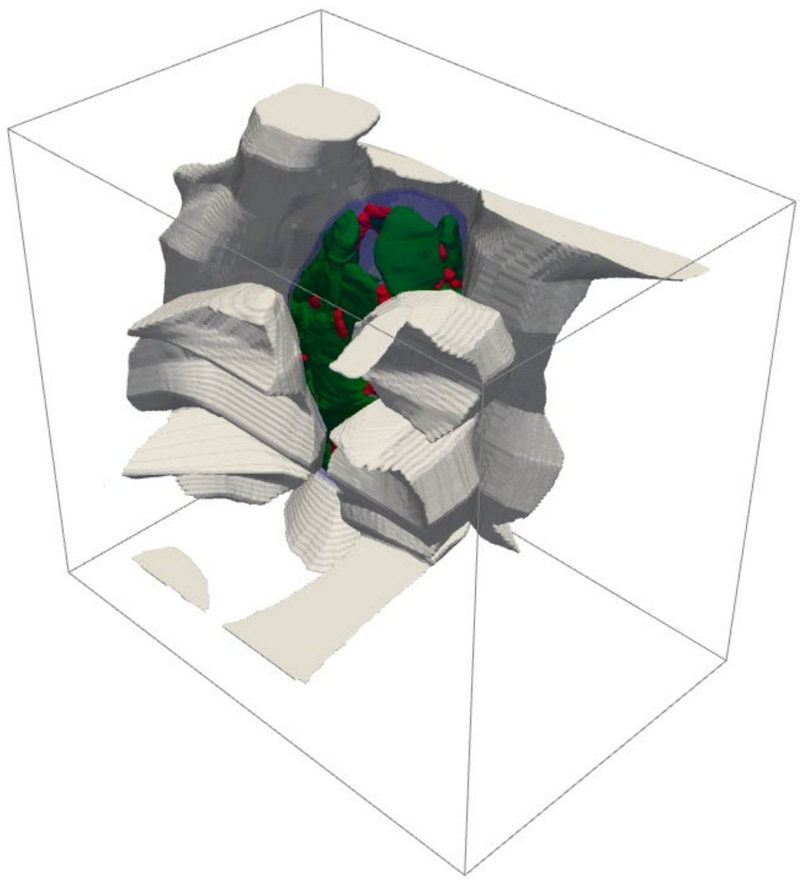
In [3]:
```
#Ideally we would read in images and convert them to meshes (3D objects) i
n Python-
#Im just doing this off an STL - the results (currently the units are wron
g so we can only do relative) should be similar
tstart = time.time()
```

In [4]:
```
os.chdir("F:/FEMPython/")

t0 = time.time()
chl= pv.read("F:/CHICKPEA MIT PROJECT/D2Cell1-This is HT/D2Cell1STLS(20nmx
50nm)/HT-CHL.stl")
#reduce the mesh quality for exploring (then change it for HQ results)
target_reduction = 0.0
chl=chl.decimate(target_reduction)
#######################################

mit= pv.read("F:/CHICKPEA MIT PROJECT/D2Cell1-This is HT/D2Cell1STLS(20nmx
50nm)/HT-MIT.stl")
mit=mit.decimate(target_reduction)
#mit=mitraw.decimate(target_reduction)
mit2=mit #wastes memory and is slow
mit3=mit

air=pv.read("F:/CHICKPEA MIT PROJECT/D2Cell1-This is HT/D2Cell1STLS(20nmx5
0nm)/HT-AIR.stl")
air=air.decimate(target_reduction)
adjcells=pv.read("F:/CHICKPEA MIT PROJECT/D2Cell1-This is HT/D2Cell1STLS(2
0nmx50nm)/HT-ADJRESHAPED.stl")
adjcells=adjcells.decimate(target_reduction)
cwproxy=adjcells+air
t1 = time.time() - t0

print('STL Load in Time: '+str(np.round(t1, 1))+' s')
```

```
STL Load in Time: 83.8 s
```

In [45]:
```
#Now its best to visualise it all

print("Chickpea Cell (HT)")



p = pv.Plotter()
p.title=("chickpea cell")
p.add_mesh(chl, color="green")
p.add_mesh(mit, color="red")
p.add_mesh(air, color="blue", opacity=0.1)
p.add_mesh(adjcells, color="white", opacity=1)
p.add_bounding_box()
p.set_background("white", top="white")
p.show()
```

```
Chickpea Cell (HT)
```



In [ ]:
```
def plot_scene_1():
    from IPython.display import display
    pv.set_plot_theme("document")
    plotter = pv.Plotter()
    plotter.add_mesh(mit, color="red")
    plotter.add_mesh(chl, color="green", opacity=1)
    plotter.add_mesh(air, color="blue", opacity=0.6)
    plotter.add_mesh(adjcells, color="white", opacity=0.6)
    disp = plotter.show(use_panel=True, auto_close=False)
```

```
    display(disp)
plot_scene_1()
```

In [7]:
```
#Now we want to get the distances from each mitochdnria to the nearest chl
oroplast
#If a mitochondria is closer there is more change for CO2 reffixation
```

In [8]:
```
t0 = time.time()
tree = KDTree(chl.points)
d, idx = tree.query(mit.points )
mit["Distance (um)"] = d/1000
t1 = time.time() - t0
print('Ktree Run Time: '+str(np.round(t1, 1))+' s')
print ('mean distance from mit to chl: '+str(np.mean(d)/1000)+ ' um')
```

```
Ktree Run Time: 1517.0 s
mean distance from mit to chl: 0.5112445031348708 um
```

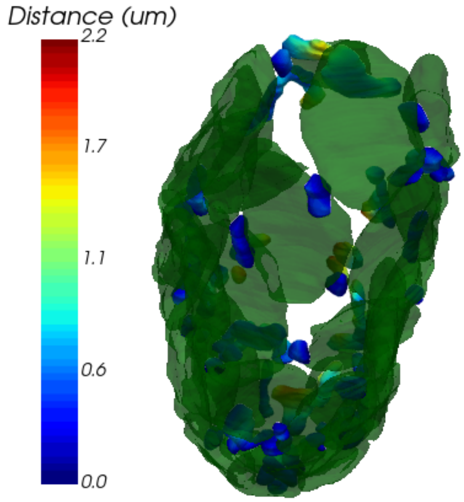
In [9]:
```
print("saving each points differences in nm")
np.savetxt("HTKDtreeMit-CHL.csv", d, delimiter=",")
```

```
saving each points differences in nm
```

In [10]:
```
#create a unique dataframe which has the mit surffcaes and there distances
 from chlroplasts
chldistances=mit
```

In [11]:
```
print("The distance between the surface of mitchondria and chlroplasts")

sargs = dict(
    title_font_size=20,
    label_font_size=15,
    shadow=True,
    n_labels=5,
    italic=True,
    fmt="%.1f",
    font_family="arial",height=0.5,
    vertical=True,
    position_x=0.3,
    position_y=0.8
)

boring_cmap = plt.cm.get_cmap("jet", 50)

pv.set_plot_theme("document")
p = pv.Plotter()
p.set_background("white")
p.add_mesh(chldistances, scalars="Distance (um)",scalar_bar_args=sargs, cm
ap=boring_cmap )
p.add_mesh(chl, color="green", opacity=0.5)
p.show()
```

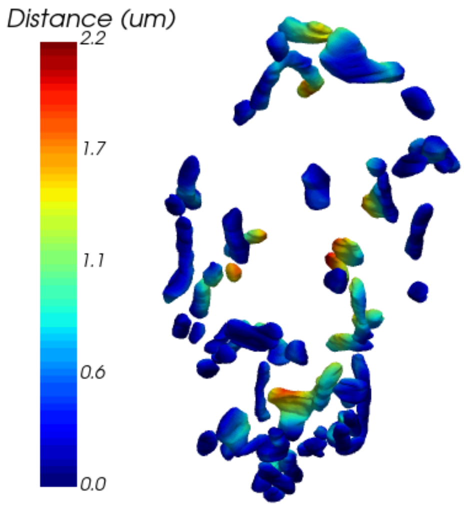
```
The distance between the surface of mitchondria and chlroplasts
```



In [12]:
```
#plotter = pv.PlotterITK()
#plotter.add_mesh(chl, color="green")
#plotter.add_mesh(mit, color="red")
#plotter.add_mesh(air, color="blue", opacity=0.1)
#plotter.add_mesh(adjcells, color="white", opacity=1)
#plotter.show(True)
```

In [13]:
```
print("The distance between the surface of mitchondria and chlroplasts")
print("(chloroplasts not shown)")
pv.set_plot_theme("document")
p = pv.Plotter()
p.add_mesh(chldistances, scalars="Distance (um)",scalar_bar_args=sargs, cm
ap=boring_cmap )
p.show()
```

```
The distance between the surface of mitchondria and chlroplasts
(chloroplasts not shown)
```



In [14]:
```
def plot_scene_1():
    from IPython.display import display
    pv.set_plot_theme("document")
    plotter = pv.Plotter()
    plotter.add_mesh(chldistances, scalars="Distance (um)",scalar_bar_args
=sargs, cmap=boring_cmap )
    plotter.add_mesh(chl, color="green", opacity=0.6)
    disp = plotter.show(use_panel=True, auto_close=False)
    display(disp)
plot_scene_1()
```

In [ ]:

In [15]:
```
t0 = time.time()
tree2 = KDTree(cwproxy.points)
d2, idx = tree2.query(mit2.points )
mit2["Distance (um)"] = d2/1000
t1 = time.time() - t0
print('Ktree Run Time: '+str(np.round(t1, 1))+' s')
print ('mean distance from mit to chl: '+str(np.mean(d2)/1000)+ ' um')
```

```
Ktree Run Time: 1912.8 s
mean distance from mit to chl: 0.9893999351763009 um
```

In [16]:
```
print("saving each points differences in nm")
np.savetxt("HTKDtreeMit-cellwall.csv", d2, delimiter=",")
```

saving each points differences in nm

In [17]:
```
#create a unique dataframe which has the mit surffcaes and there distances
 from cellwall
cwdistances=mit2
```

In [ ]:
```
```

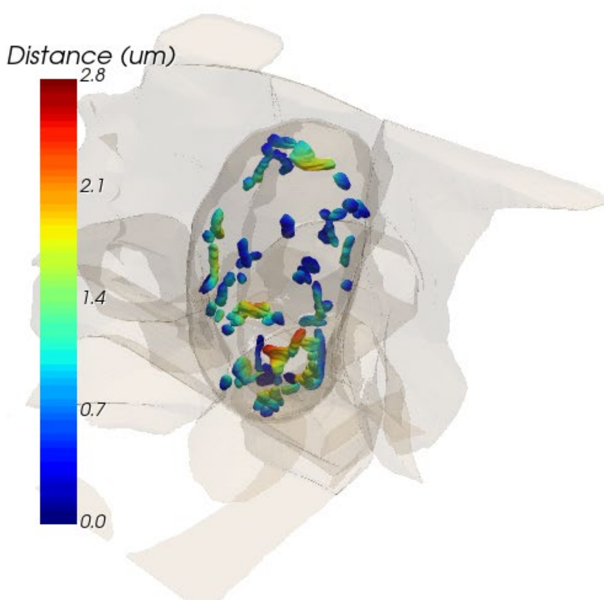
In [18]:
```
#So on average mitochondria are closer to the cell wall.(stress this is ju
st one celll)
print("the difference between distances")
np.mean(d2 - d)/1000
```

```
the difference between distances
```

Out[18]: 0.4781554320414302

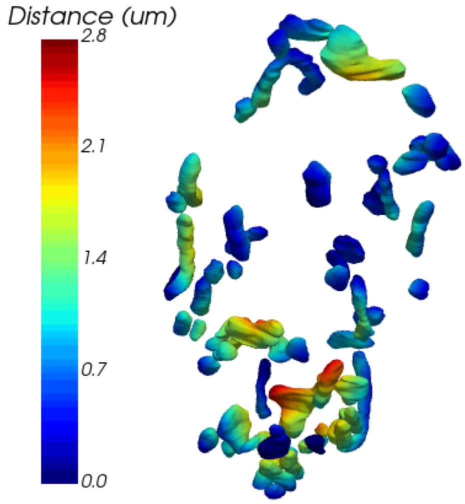
In [19]:
```
print("The distance between the surface of mitchondria and cell walls")
pv.set_plot_theme("document")
p = pv.Plotter()
p.add_mesh(cwdistances, scalars="Distance (um)",scalar_bar_args=sargs, cma
p=boring_cmap )
p.add_mesh(cwproxy, color=True, opacity=0.1)
p.show()
```

```
The distance between the surface of mitchondria and cell walls
```



In [20]:
```
print("The distance between the surface of mitchondria and cellwalls")
print("The Cell Wall is removed")
pv.set_plot_theme("document")
p = pv.Plotter()
p.add_mesh(cwdistances, scalars="Distance (um)",scalar_bar_args=sargs, cma
p=boring_cmap)
p.show()
```

```
The distance between the surface of mitchondria and cellwalls
The Cell Wall is removed
```



In [21]:
```
def plot_scene_1():
    from IPython.display import display
    pv.set_plot_theme("document")
    plotter = pv.Plotter()
    plotter.add_mesh(cwdistances, scalars="Distance (um)",scalar_bar_args=
sargs, cmap=boring_cmap )
    plotter.add_mesh(cwproxy, color=True, opacity=0.1)
    disp = plotter.show(use_panel=True, auto_close=False)
    display(disp)
plot_scene_1()
```

In [22]:
```
#Now for every point on the mitchdnria we have
#1) the distance to the nearest chorloplast
#2) the distance to the nearest cell wall

# so we can calculate the difference for each point (cell wall distance -
```

```
chlroplast distance)
# a positive number means the chlroplast is closer
# a negative number means the cell wall is closer.
```

In [ ]:

```
```

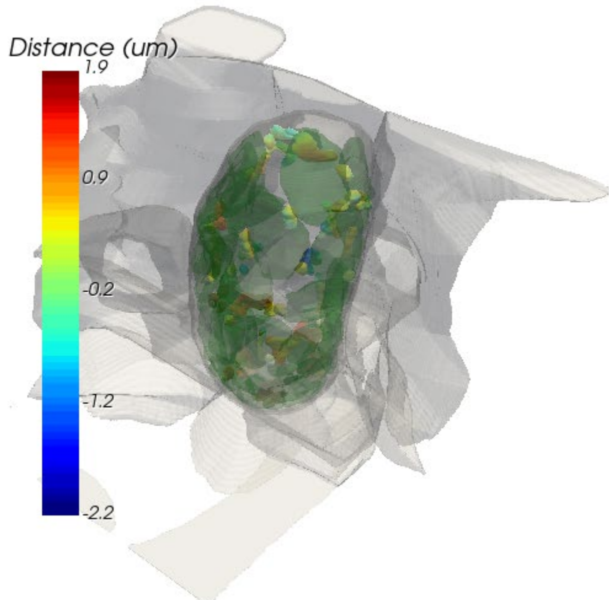
In [23]:

```
#Create a new object that has the difference instead of the actual distanc
e
cwdif=mit3
dif=d2-d
cwdif["Distance (um)"]=dif/1000
```

In [24]:

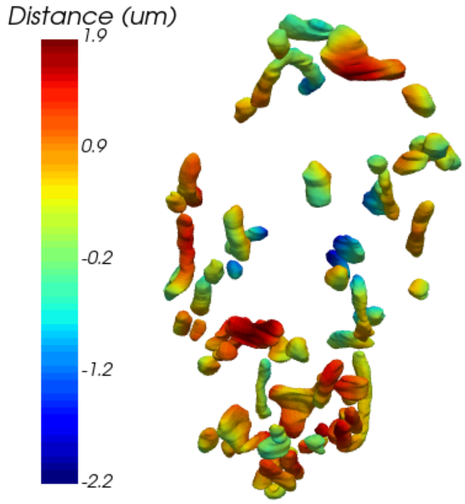
```
print("The difference between the two mesurments (cell wall distance - chl
 distance)")
print("darker colours (negative) are where mironcdhria are closer to the c
ell")
print("warmer colours (positive) are where mironcdhria are closer to the c
hlroplast")
pv.set_plot_theme("document")
p = pv.Plotter()
p.add_mesh(cwdif, scalars="Distance (um)",scalar_bar_args=sargs, cmap=bori
ng_cmap )
p.add_mesh(cwproxy, color="white", opacity=0.3)
p.add_mesh(chl, color="green", opacity=0.3)
#p.add_bounding_box()
#p.set_background("white", top="white")
p.show()
```

```
The difference between the two mesurments (cell wall distance - chl distan
ce)
darker colours (negative) are where mironcdhria are closer to the cell
warmer colours (positive) are where mironcdhria are closer to the chlropla
st
```



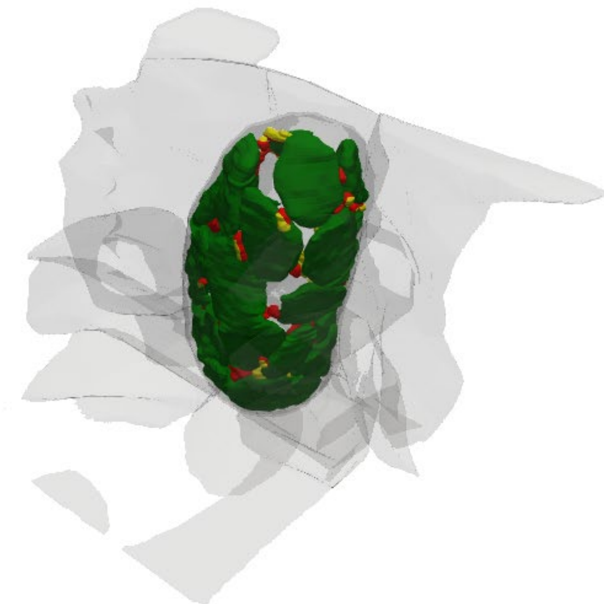
In [25]:
```
print("The difference between the two mesurments (cell wall distance - chl
 distance)")
print("cell wall and chlroplast removed")
pv.set_plot_theme("document")
p = pv.Plotter()
p.add_mesh(cwdif, scalars="Distance (um)",scalar_bar_args=sargs, cmap=bori
ng_cmap )
p.show()
```

```
The difference between the two mesurments (cell wall distance - chl distan
ce)
cell wall and chlroplast removed
```



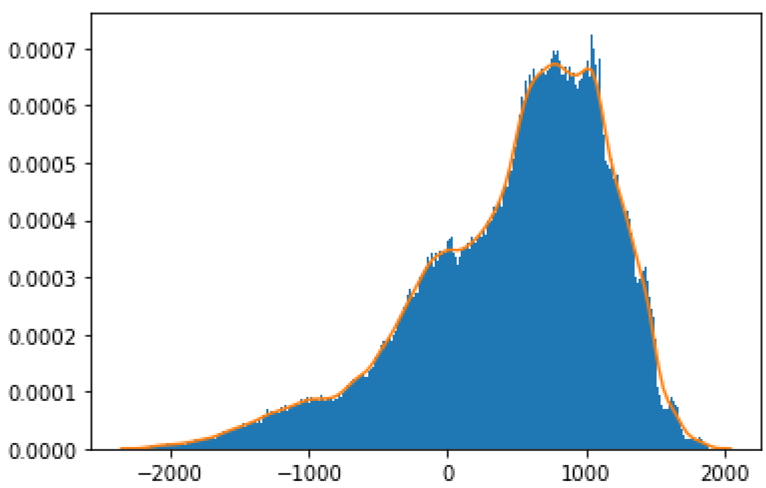
In [26]:
```
print("The difference between the two mesurments (cell wall distance - chl
 distance)")
print("This is Binary= The warmer colours (where mit are closer to chl tur
n red")
print("and the cooler colours (wheremit are closer to cell wall turn yello
w")

scalars = np.empty(cwdif.n_points)
scalars[cwdif['Distance (um)'] < 0] = 4  # red
scalars[cwdif['Distance (um)'] > 0] = 2  # yellow
pv.set_plot_theme("document")
p = pv.Plotter()
p.add_mesh(cwdif, scalars=scalars, cmap=['red', 'yellow'])
p.add_mesh(cwproxy, color="grey", opacity=0.1)
p.add_mesh(chl, color="green", opacity=1)

#p.add_bounding_box()
#p.set_background("white", top="white")
p.show()
```

```
The difference between the two mesurments (cell wall distance - chl distan
ce)
This is Binary= The warmer colours (where mit are closer to chl turn red
and the cooler colours (wheremit are closer to cell wall turn yellow
```



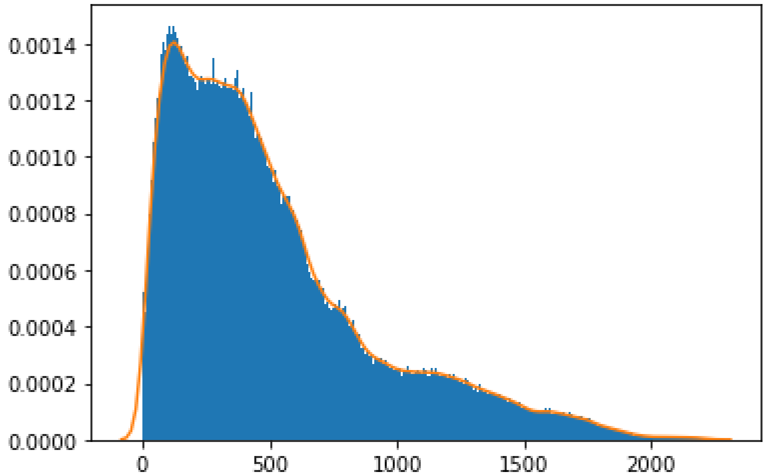
In [27]:
```
plt.hist(dif, density=True, bins=300)  # This is difference
sns.kdeplot(data=dif)
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1d21c898d60>



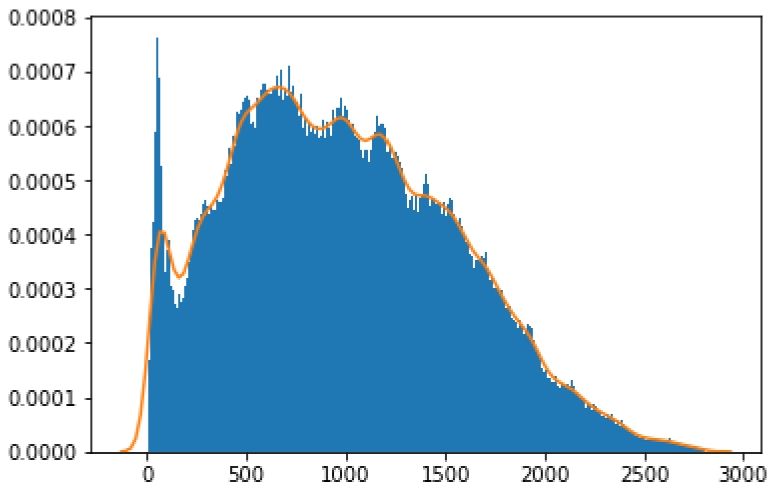
In [28]:
```
plt.hist(d, density=True, bins=300) # mit to chl distance
sns.kdeplot(data=d)
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1d184896730>



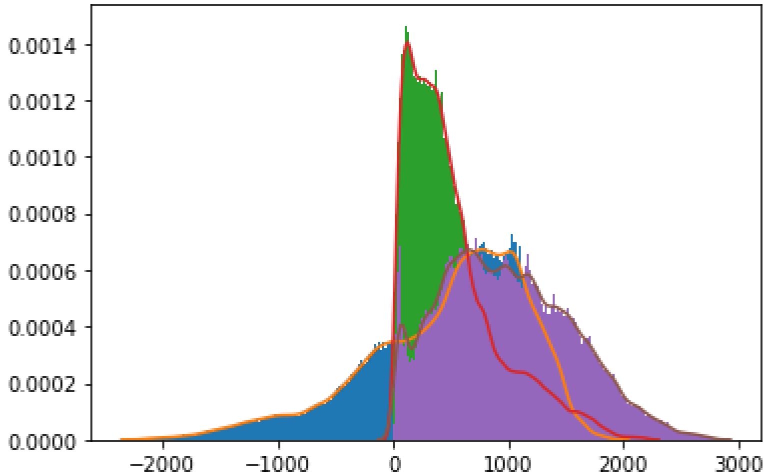
In [29]:
```
plt.hist(d2, density=True, bins=300) # mit to cellwall distance
sns.kdeplot(data=d2)
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1d184b81820>



In [30]:
```
plt.hist(dif, density=True, bins=300)  # `density=False` would make counts
sns.kdeplot(data=dif)
plt.hist(d, density=True, bins=300)  # `density=False` would make counts
sns.kdeplot(data=d)
plt.hist(d2, density=True, bins=300)  # `density=False` would make counts
sns.kdeplot(data=d2)
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1d18516e8e0>



In [31]:
```
##########################################################################
##
###########MCHL DISTANCE TO IAS ################################
```

In [32]:
```
t0 = time.time()
chl1=chl
tree3 = KDTree(air.points)
d3, idx = tree3.query(chl1.points )
chl1["Distance (um)"] = d3/1000
t1 = time.time() - t0
print('Ktree Run Time: '+str(np.round(t1, 1))+' s')
print ('mean distance from chl to air: '+str(np.mean(d3)/1000)+ ' um')
```

```
Ktree Run Time: 8024.3 s
mean distance from chl to air: 0.9796673605552247 um
```

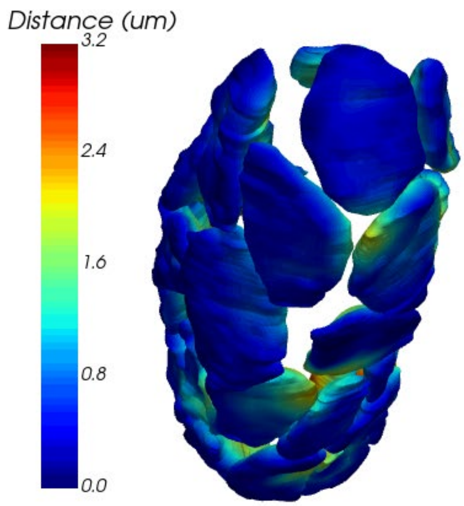
In [33]:
```
print("saving each points differences in nm")
np.savetxt("HTKDtreechl-air.csv", d, delimiter=",")
```

```
saving each points differences in nm
```

In [34]:
```
sc=chl1
```

In [35]:
```
print("The distance between the surface of chl and air")
pv.set_plot_theme("document")
p = pv.Plotter()
p.add_mesh(sc, scalars="Distance (um)",scalar_bar_args=sargs, cmap=boring_
cmap )
#p.add_mesh(air, color="grey", opacity=0.7)
p.show()
```

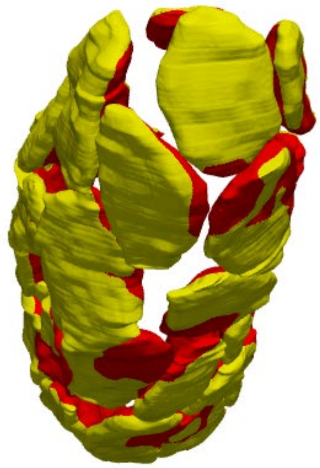
```
The distance between the surface of chl and air
```



In [36]:

```
print("This is Binary= The warmer colours (where chls are closer to IAS tu
rn red")
print("and the cooler colours (where chl are closer to cell wall turn yell
ow")
pv.set_plot_theme("document")
p = pv.Plotter()
scalars = np.empty(sc.n_points)
scalars[sc['Distance (um)'] < 0.75] = 4  # red
scalars[sc['Distance (um)'] > 0.75] = 2  # yellow
pv.set_plot_theme("document")
p = pv.Plotter()
p.add_mesh(sc, scalars=scalars, cmap=['red', 'yellow'])
#p.add_mesh(cwproxy, color="grey", opacity=0.1)
#p.add_mesh(chl, color="green", opacity=1)
p.show()
```

```
This is Binary= The warmer colours (where chls are closer to IAS turn red
and the cooler colours (where chl are closer to cell wall turn yellow
```



In [37]:
```
def plot_scene_1():
    from IPython.display import display
    pv.set_plot_theme("document")
    plotter = pv.Plotter()
    plotter.add_mesh(sc, scalars=scalars, cmap=['red', 'yellow'])
    plotter.add_mesh(air, color="blue", opacity=0.5)
    disp = plotter.show(use_panel=True, auto_close=False)
    display(disp)
plot_scene_1()
```

In [38]:
```
t1 = time.time() - tstart
print('Total Run Time: '+str(np.round(t1, 1))+' s')
```

```
Total Run Time: 11653.3 s
```

In [42]:
```
#Lets get some last information
mit
```

Out[42]:

Header	Data Arrays

PolyData	Information
N Cells	984724
N Points	492460
X Bounds	3.366e+04, 4.571e+04

Name	Field	Type	N Comp	Min	Max
Distance (um)	Points	float64	1	-2.198e+00	1.887e+00

Y Bounds	6.709e+03, 2.205e+04
Z Bounds	3.825e+03, 2.972e+04
N Arrays	2

Data	Points	float64	1	2.000e+00	4.000e+00

```
In [43]: chl
```

Out[43]:

Header **Data Arrays**

PolyData	Information
N Cells	5131972
N Points	2565947
X Bounds	3.318e+04, 4.630e+04
Y Bounds	6.474e+03, 2.216e+04
Z Bounds	3.375e+03, 3.052e+04
N Arrays	2

Name	Field	Type	N Comp	Min	Max
Distance (um)	Points	float64	1	5.527e-03	3.201e+00
Data	Points	float64	1	2.000e+00	4.000e+00

```
In [ ]:
```