paper_figures

May 25, 2022

[1]:
```
import sys
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import collections as mc
from mpl_toolkits.axes_grid1 import make_axes_locatable
import os
import pathlib
import random
import importlib as imp
import scipy.stats as stats
import scipy.optimize as opt
import itertools as iter
import math
import ipywidgets
from tqdm.notebook import tqdm, trange
import datetime
import time
import json
import h5py
import lmfit
import pandas as pd
import datetime
import psutil

import flow_fields as flow
import charge_transfer_assets as assets
for mod in [flow, assets]:
    imp.reload(mod) # Updates any changes made to the imported scripts

# Set matplotlib format
SMALL_SIZE = 18
MEDIUM_SIZE = 20
BIGGER_SIZE = 22

plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
```

```
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
plt.rc('figure', titlesize=BIGGER_SIZE)  # fontsize of the figure title
```

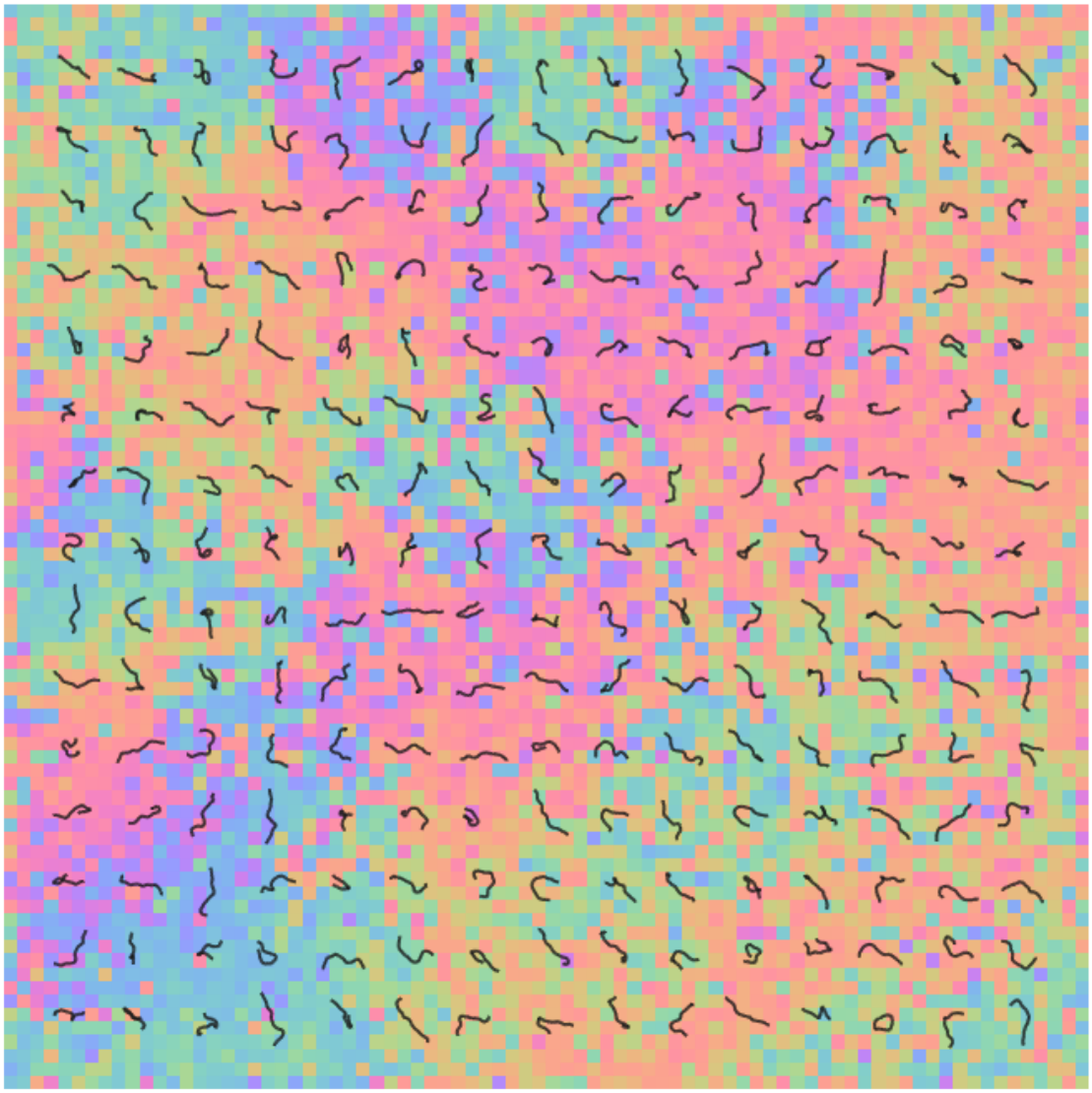
[2]:
```
print('RAM memory % used:', psutil.virtual_memory()[2])
print('RAM memory Available: %.1f GB' % (int(psutil.virtual_memory()[1]) / 10.
 ↪0**9))
```

```
RAM memory % used: 57.5
RAM memory Available: 42.9 GB
```

1 Figure 2b

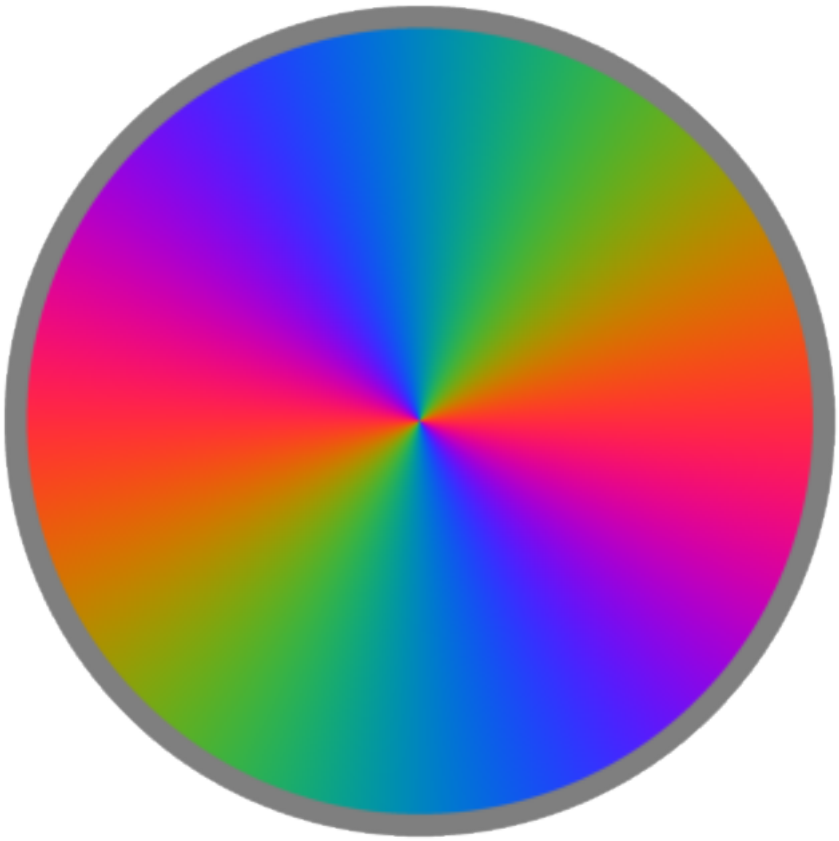
[3]:
```
imp.reload(assets)  # Updates any changes made to plotting functions

xyz_path = '../data/chain_geometries'
files = np.array([f for f in os.listdir(xyz_path) if 'r50v' in f])
field_strengths = [int(f.split('v')[1]) for f in files]
sorted_files = files[np.argsort(field_strengths)]
sorted_field_strengths = np.sort(field_strengths)
step_size = 10  # nm
file_number = 10
shape = ((80, 80, 1))
chain_length = 54  # nm

xyz, xyz_full_path = assets.get_xyz(xyz_path, sorted_files, file_number,␣
 ↪step_size, rotate=True)
chains = assets.ChainSet(xyz, chain_length, xyz_full_path)
chains.create_bins(shape)

chains.create_bins(shape)
chains_image = chains.plot_lines(grid=(15, 15), linewidth=2, method='Color by␣
 ↪Bin', alpha=0.5)
chains_image.savefig('../figures/Chains_Image.png', dpi=300,␣
 ↪bbox_inches='tight')
```

```
Chosen File:  r50v11
```



2 Color Wheel

[4]:
```
plt.figure(figsize=(6, 6))
color_wheel = assets.plot_color_wheel(length=2000)
color_wheel.savefig('../figures/color_wheel.png', dpi=300, transparent=True,␣
↪bbox_inches='tight')
```



3 Figure 3a

[5]:
```
# Detailed rate data is not saved after a simulation, so it will be re-computed
 ↪here.  This may take a few minutes.
xyz_path = '../data/chain_geometries'
files = np.array([f for f in os.listdir(xyz_path) if 'r50v' in f])
field_strengths = [int(f.split('v')[1]) for f in files]
sorted_files = files[np.argsort(field_strengths)]
sorted_field_strengths = np.sort(field_strengths)
step_size = 10  # nm
file_number = 10
shape = ((80, 80, 1))
chain_length = 54  # nm

xyz, xyz_full_path = assets.get_xyz(xyz_path, sorted_files, file_number,
 ↪step_size, rotate=True)
chains = assets.ChainSet(xyz, chain_length, xyz_full_path)
```

```
chains.create_bins(shape)
new = assets.PathExperiment()
new.compute_rates(chains, 10**6, 0, n_times=5, images=False)
```

```
Chosen File:  r50v11
Estimated rates: On-chain = 8.90E+10 per second, Interchain = 5.26E+06 per
second
Diagonalizing Rates…
Pre-computing matrix exponentials for Times 2.86E-13 to 4.87E-08 s
Expected RAM use 316.32768 MB
```

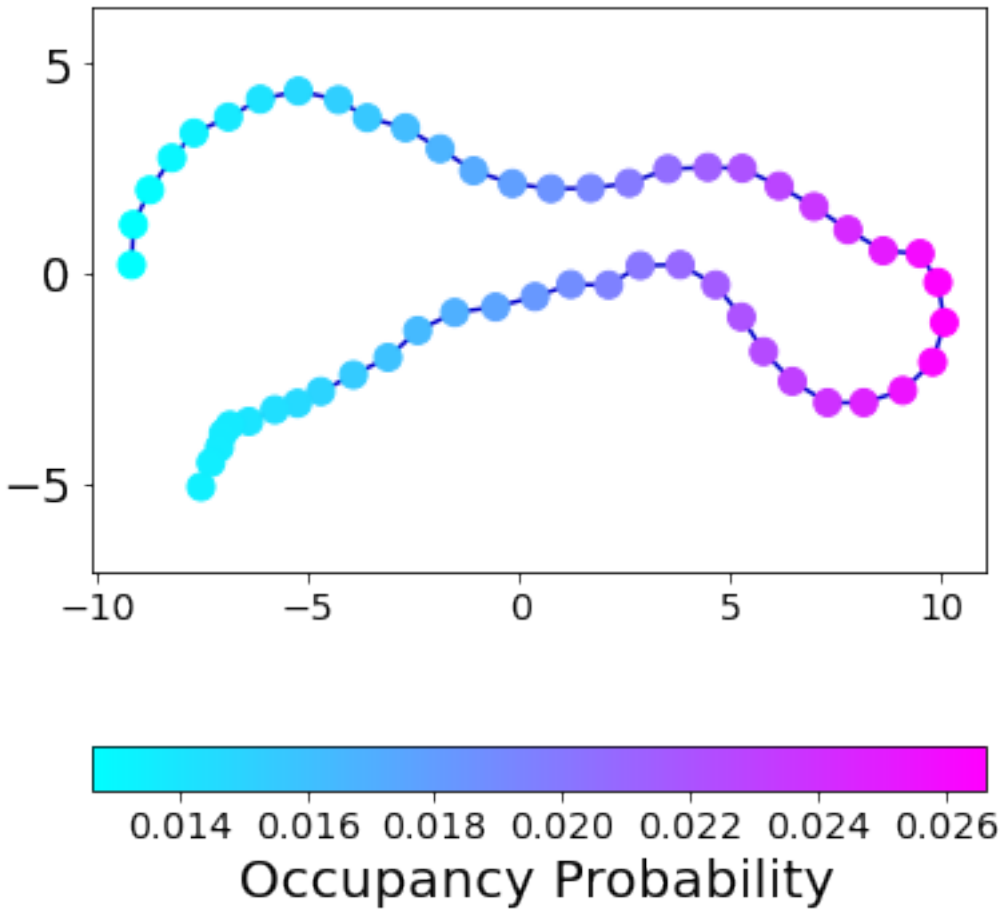
```
Exponentiating Matrices…:   0%|          | 0/10848 [00:00<?, ?it/s]
```

```
Recording Short-Time Mobilities…
```

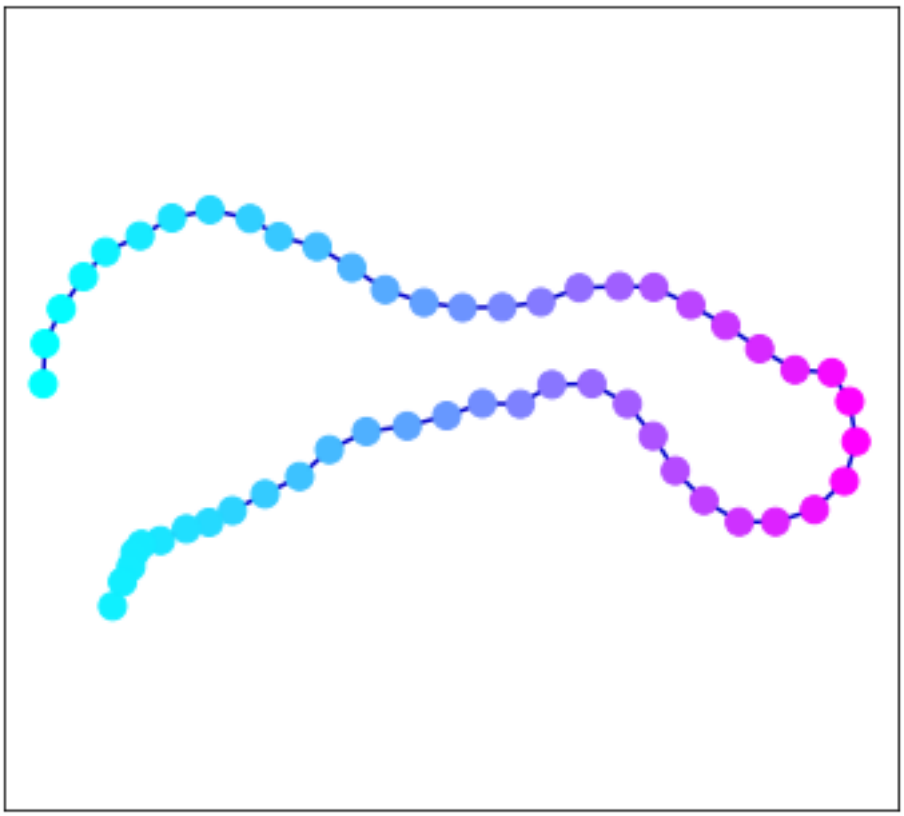
[6]:
```
imp.reload(assets)

# Change some formatting away from defaults
plt.rc('xtick', labelsize=14)    # fontsize of the tick labels


plt.figure(figsize=(6, 6))
# First Version to get Colorbar the correct size
folded_chain = assets.PathExperiment.plot_chain_probability_distribution(new,␣
 ↪histogram=False, n_chains=1, start=3711, show=False)
folded_chain_size_1 = plt.gcf()
plt.show()

# Second version to get frame the correct size
plt.figure(figsize=(6, 6))
folded_chain = assets.PathExperiment.plot_chain_probability_distribution(new,␣
 ↪histogram=False, n_chains=1, start=3711, show=False, colorbar=False)
plt.gca().set_box_aspect(0.9) # Aspect ratio is still equal to 1
plt.gca().xaxis.set_visible(False)
plt.gca().yaxis.set_visible(False)
folded_chain_size_2 = plt.gcf()
plt.show()

# Set formatting back to defaults
plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
plt.rc('figure', titlesize=BIGGER_SIZE)  # fontsize of the figure title
```

```
Chain Number 3711 ^^^
```

Potential Difference 19 mV
-9.2 to 10.1
Probability Ratio 2.1



Chain Number 3711 ^^^
Potential Difference 19 mV
-9.2 to 10.1
Probability Ratio 2.1



[7]:
```
folded_chain_size_1.savefig('../figures/folded_chain_size_1.png', dpi=300)
folded_chain_size_2.savefig('../figures/folded_chain_size_2.png', dpi=300)
```

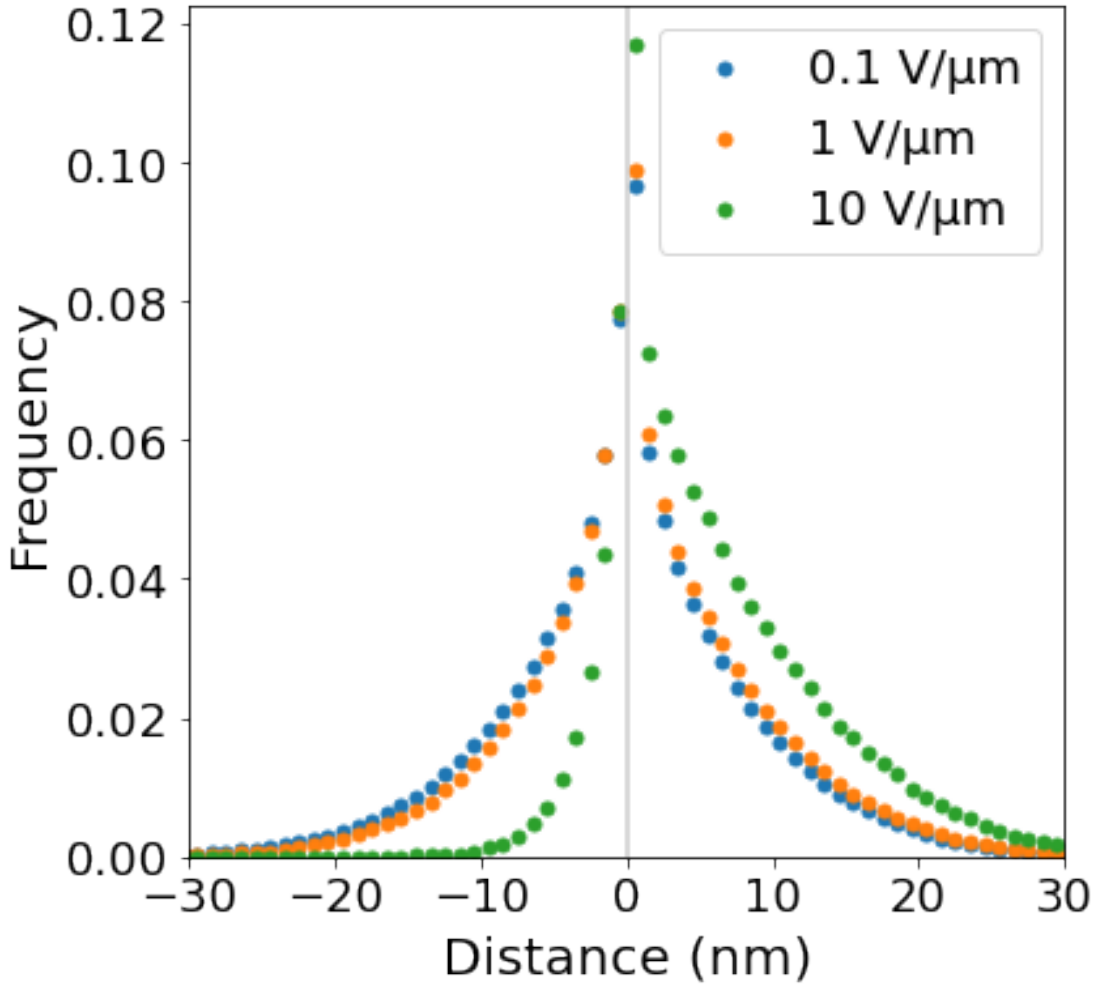
4 Figure 3 b-c

[8]:
```
imp.reload(assets)
file_manager = assets.FileManager('../data/transport_simulations/')
file_manager.add_files('ct_29_n100/', angle=90)
```

```
File Added: 1 Total Files
File Added: 2 Total Files
File Added: 3 Total Files
RAM memory % used: 60.5
RAM memory Available: 39.9 GB
```

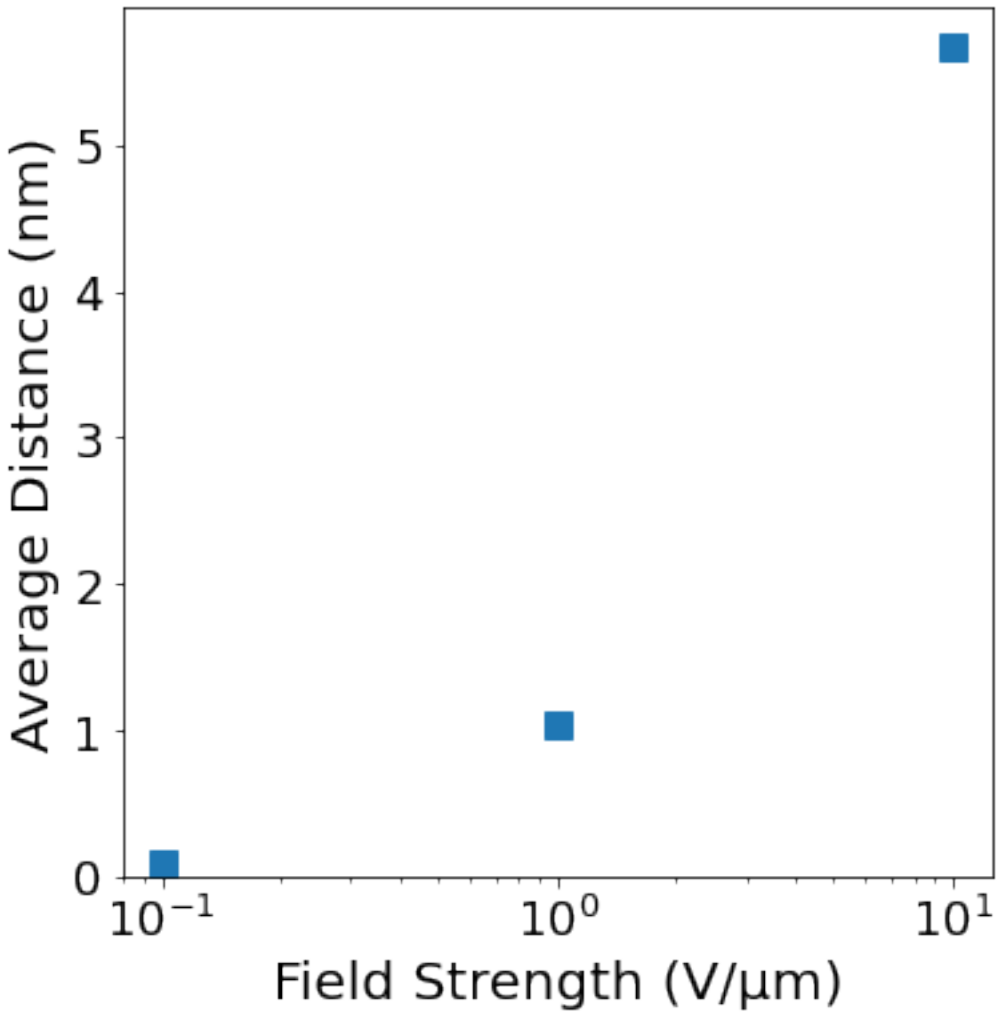
[9]:
```
# Frame b
plt.figure(figsize=(6, 6))
for f in file_manager.files[::-1]:
```

```
    if f.field in [10**7, 10**6, 10**5]:
        plt.plot(f.move_distances, f.move_frequencies / np.sum(f.
↪move_frequencies), marker='.', linewidth=0, markersize=10)
plt.xlim(-30, 30)
plt.ylim(0)
plt.axvline(x=0, c=(0.8, 0.8, 0.8))
plt.legend(['0.1 V/\u03bcm', '1 V/\u03bcm', '10 V/\u03bcm'])
plt.xlabel('Distance (nm)')
plt.ylabel('Frequency')
distance_histogram = plt.gcf()
plt.show()
```



[10]:
```
# Frame c

plt.figure(figsize=(6, 6))
x = [0.1, 1, 10]
```

```
y = [f.avg_forward_distance for f in file_manager.files][::-1]
plt.plot(x, y, marker='s', linewidth=0, markersize=10)
plt.xscale('log')
plt.xlabel('Field Strength (V/\u03BCm)')
plt.ylabel('Average Distance (nm)')
plt.ylim(0)
plt.gca().set_box_aspect(1)
avg_distance_vs_field = plt.gcf()

plt.show()
```



[11]:
```
distance_histogram.savefig('../figures/Distance_Histogram.png', dpi=300,␣
↪bbox_inches='tight')
avg_distance_vs_field.savefig('../figures/Average_Distance_Vs_Field.png',␣
↪dpi=300, bbox_inches='tight')
```

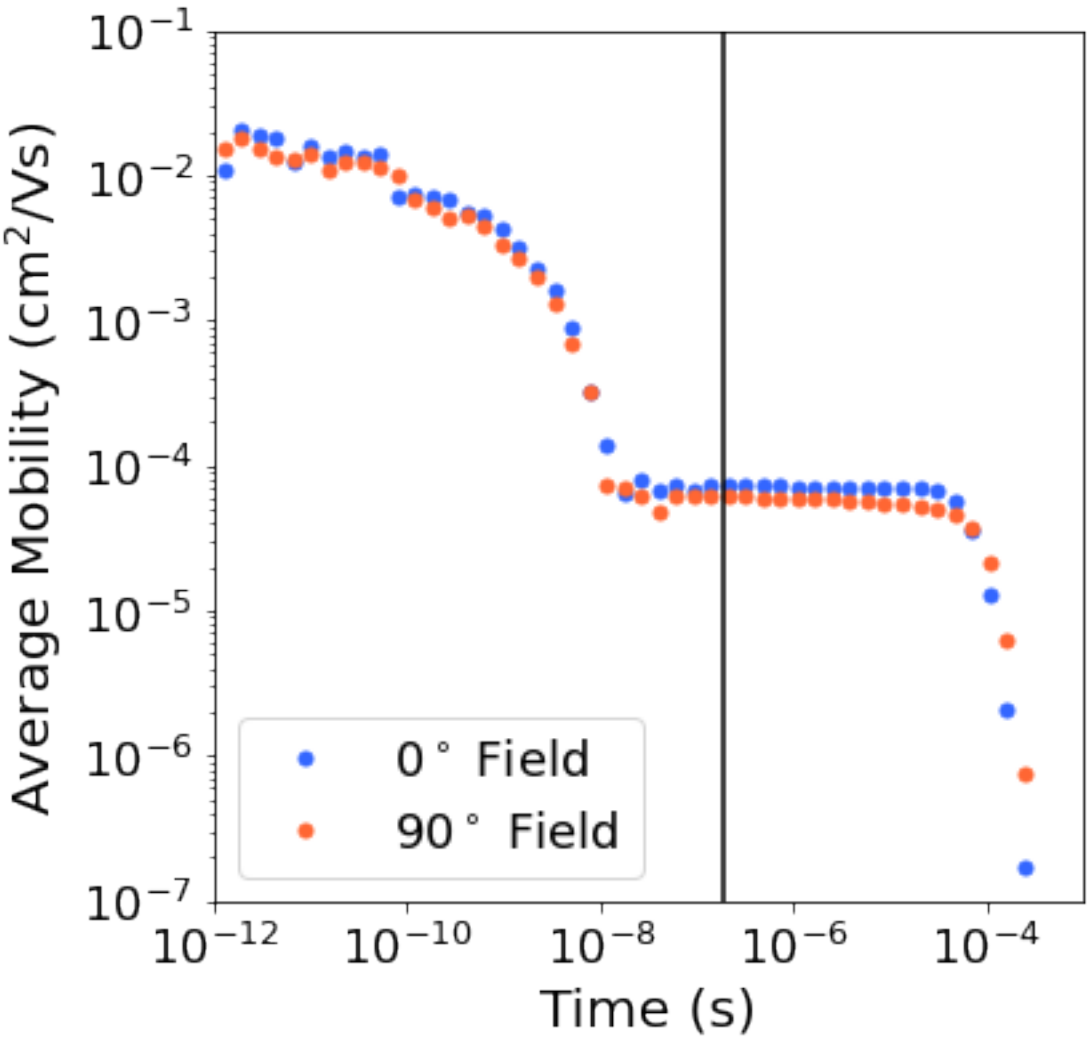
5 Figure 4a

[12]:
```
imp.reload(assets)
file_manager = assets.FileManager('../data/transport_simulations/')
file_manager.add_files('ct_30_n10000/')
```

```
File Added: 1 Total Files
File Added: 2 Total Files
RAM memory % used: 60.9
RAM memory Available: 39.5 GB
```

[13]:
```
short_time_mobilities = np.zeros(2)
long_time_mobilities = np.zeros(2)
plt.figure(figsize=(6, 6))

for e in file_manager.files:

    if e.angle == 0:
        color = (0.2, 0.4, 1)
    else:
        color = (1, 0.4, 0.2)

    # Set up reused variables
    n_blocks, n_charges = e.positions_with_time.shape
    end = e.completion_block if e.completion_block < n_blocks else e.
↪completion_block - 1
    start = 0

    short_mobility = e.get_plateau_mobility(t_max=(1/e.interchain_rate))
    long_mobility = e.get_plateau_mobility(t_min=(1/e.interchain_rate), t_max=e.
↪time_seconds[np.min(e.finish_blocks[np.nonzero(e.finish_blocks)])])
    index = 0 if e.angle == 0 else 1
    short_time_mobilities[index] = short_mobility
    long_time_mobilities[index] = long_mobility

    time_centers = ((e.time_seconds[1:] + e.time_seconds[:-1]) / 2)
    time_values = time_centers[start:end]

    field_vpcm = e.field / 10**2
    velocity_values = e.avg_velocity[start:end]
    mobility_values = velocity_values / field_vpcm  * e.sign

    plt.plot(time_values, mobility_values, linewidth=0, marker='.',␣
↪markersize=10, c=color, label='$%d^\circ$ Field ' % e.angle)
```

```
plt.xlim(10**-12, 10**-3)
plt.ylim(10**-7, 10**-1)
plt.gca().set_aspect(9/6)

plt.axvline(x=1/e.interchain_rate, c=(0, 0, 0))

plt.xlabel('Time (s)')
plt.ylabel('Time (s)')
plt.ylabel('Average Mobility (cm$^2$/Vs)')
plt.xscale('log')
plt.yscale('log')

plt.legend()
mobility_vs_time = plt.gcf()

plt.show()

print('Short Time Anisotropy Ratio: %.2f' % (short_time_mobilities[0] /␣
 ↪short_time_mobilities[1]))
print('Long Time Anisotropy Ratio: %.2f' % (long_time_mobilities[0] /␣
 ↪long_time_mobilities[1]))
```




```
Short Time Anisotropy Ratio: 1.18
Long Time Anisotropy Ratio: 1.26
```

[14]:
```
mobility_vs_time.savefig('../figures/Mobility_Vs_Time.png', dpi=300,␣
↪bbox_inches='tight')
```

6 Figure 4b-c

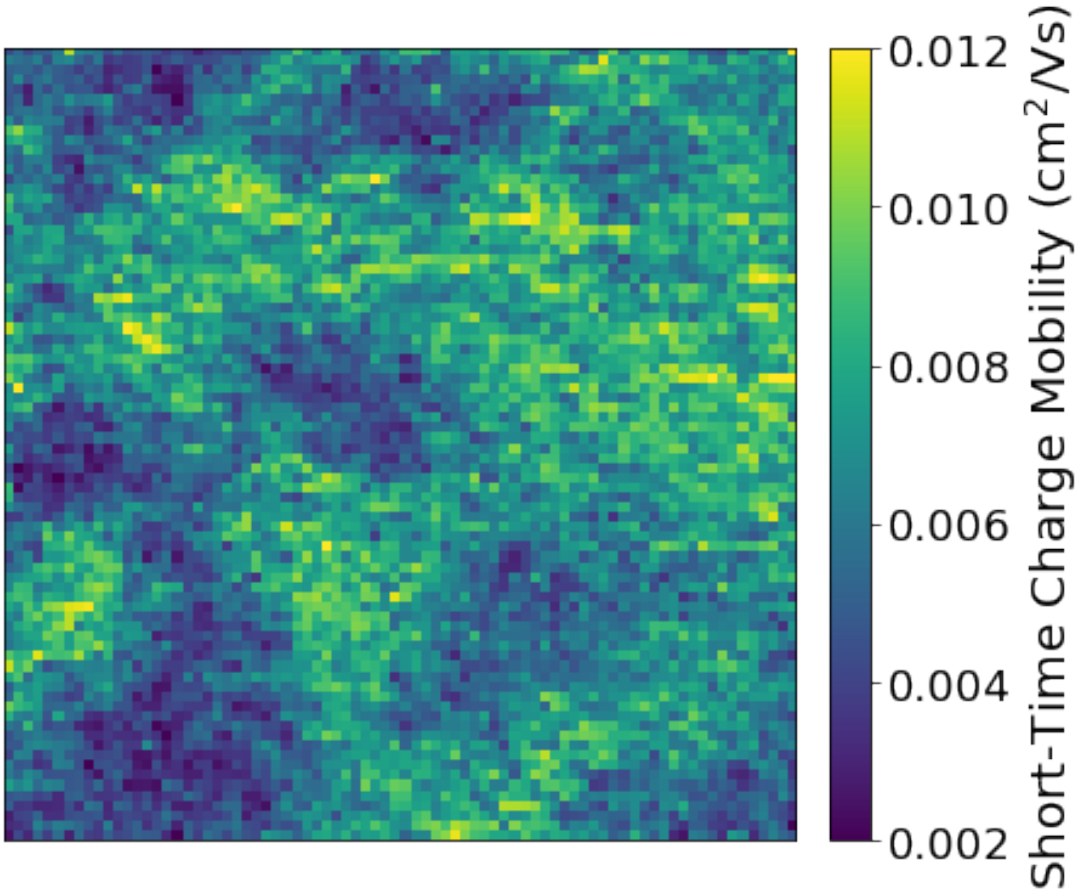
[15]:
```
file_manager = assets.FileManager('../data/transport_simulations/')
file_manager.add_files('ct_30_n10000/')
```

```
File Added: 1 Total Files
File Added: 2 Total Files
```

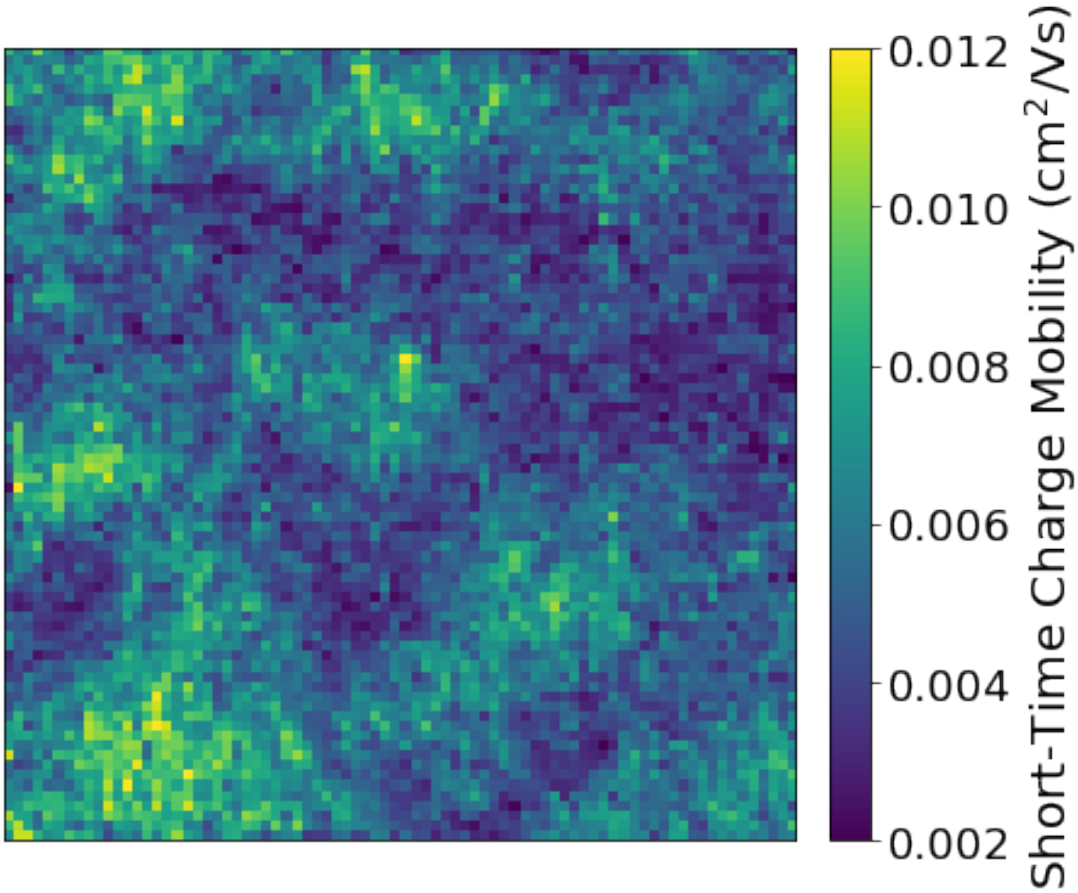
```
RAM memory % used: 61.5
RAM memory Available: 38.8 GB
```

[16]:
```
vmin = 0.002
vmax = 0.012

for angle in [0, 90]:
    experiment = file_manager.get_files_where(angle=angle)[0]
    print(experiment.angle)

    plt.figure(figsize=(8, 6))
    # Include one with colorbar
    im = assets.consistent_imshow(experiment.ns_mobility, vmin=vmin, vmax=vmax)
    plt.gca().xaxis.set_visible(False)
    plt.gca().yaxis.set_visible(False)

    colorbar = True
    if colorbar:
        ax = plt.gca()
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="5%", pad=0.2)
        cbar = plt.colorbar(im, cax=cax)
        cbar.set_label(label='Short-Time Charge Mobility (cm$^2$/Vs)')

    plt.savefig('../figures/ns_mobility_%d_degrees.png' % angle, dpi=300,␣
↪bbox_inches='tight')

    plt.show()
```

```
0
```


0.012
0.010
0.008
0.006
0.004
0.002
Short-Time Charge Mobility (cm²/Vs)

90



7 Figure 5a-g

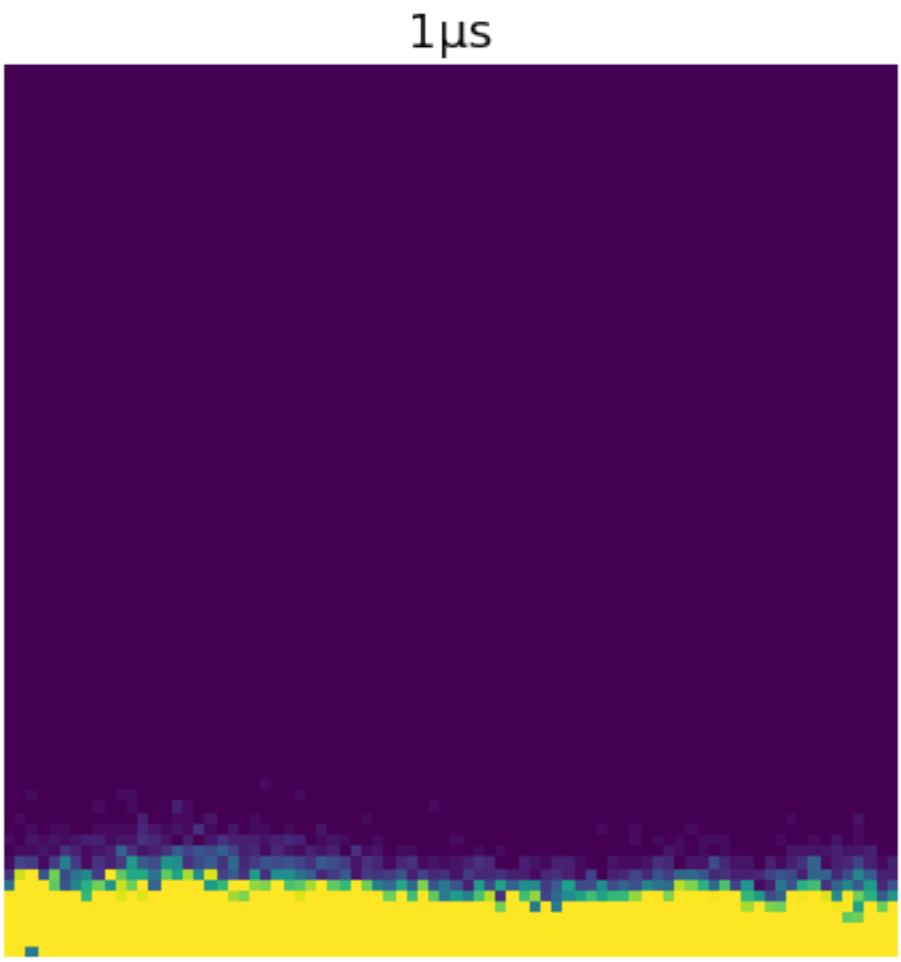
[17]:
```
imp.reload(assets)
file_manager = assets.FileManager('../data/transport_simulations/')
file_manager.add_files('ct_28_n1000/', angle=90, field=10**6)
e = file_manager.files[0]
```

```
File Added: 1 Total Files
RAM memory % used: 61.5
RAM memory Available: 38.8 GB
```

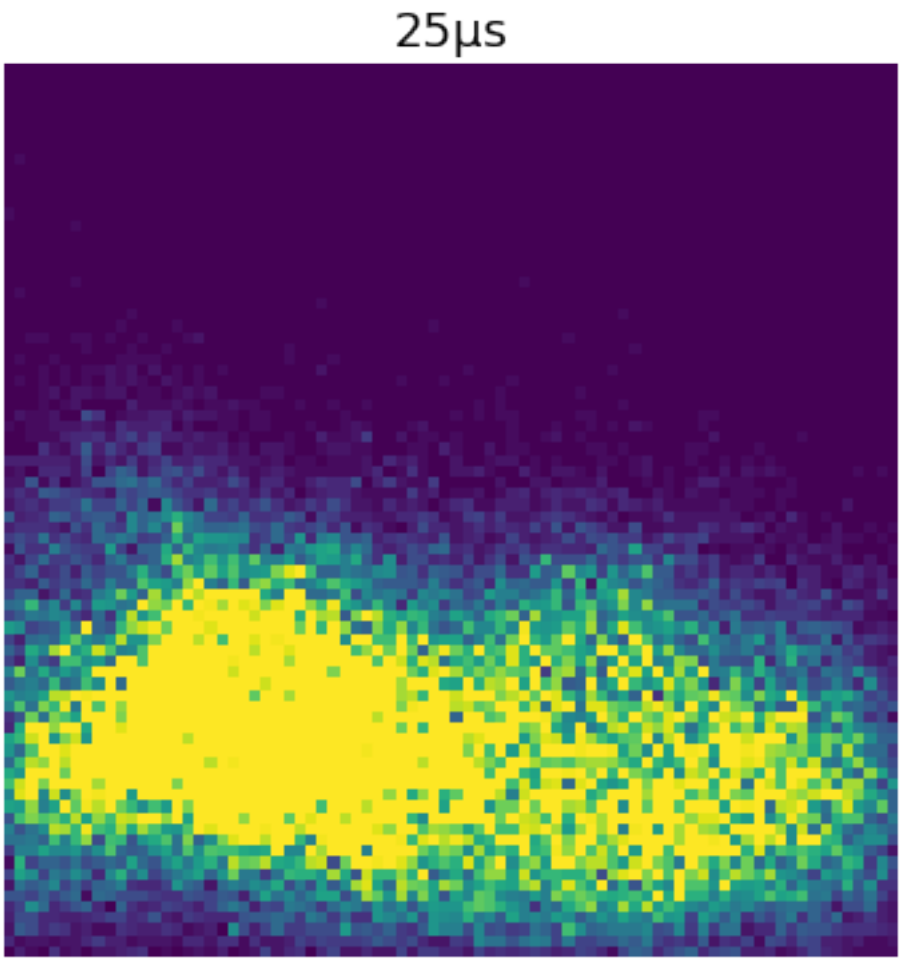
[18]:
```
images = []
# times = np.array([20, 100, 200, 300, 400, 500])*10**-6
times = np.array([1, 25, 50, 75, 100, 125])*10**-6
for t in times:
    plt.figure(figsize=(6, 8))
```

```
    new = e.plot_density_specific_time(t, normalize_by_total_mean=True,␣
↪colorbar=False, vmax=10, title=True)
    images.append(new)
```

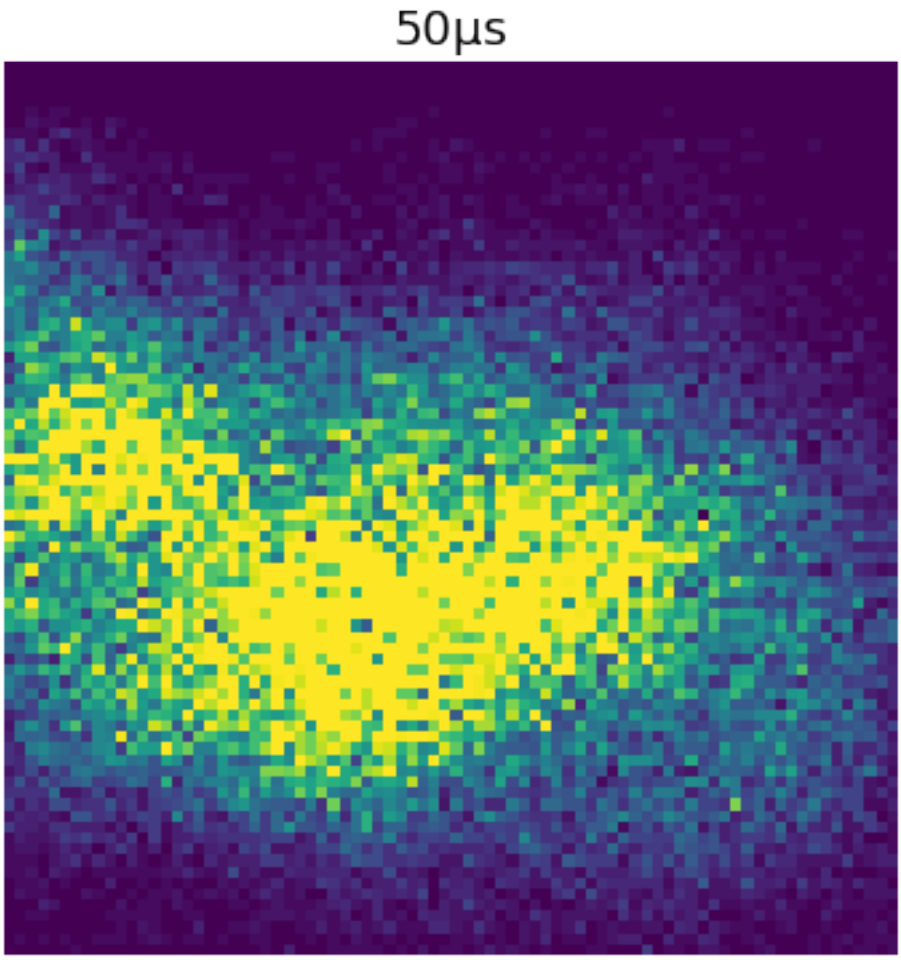
```
    Time=1.0  s
-0% Error
```



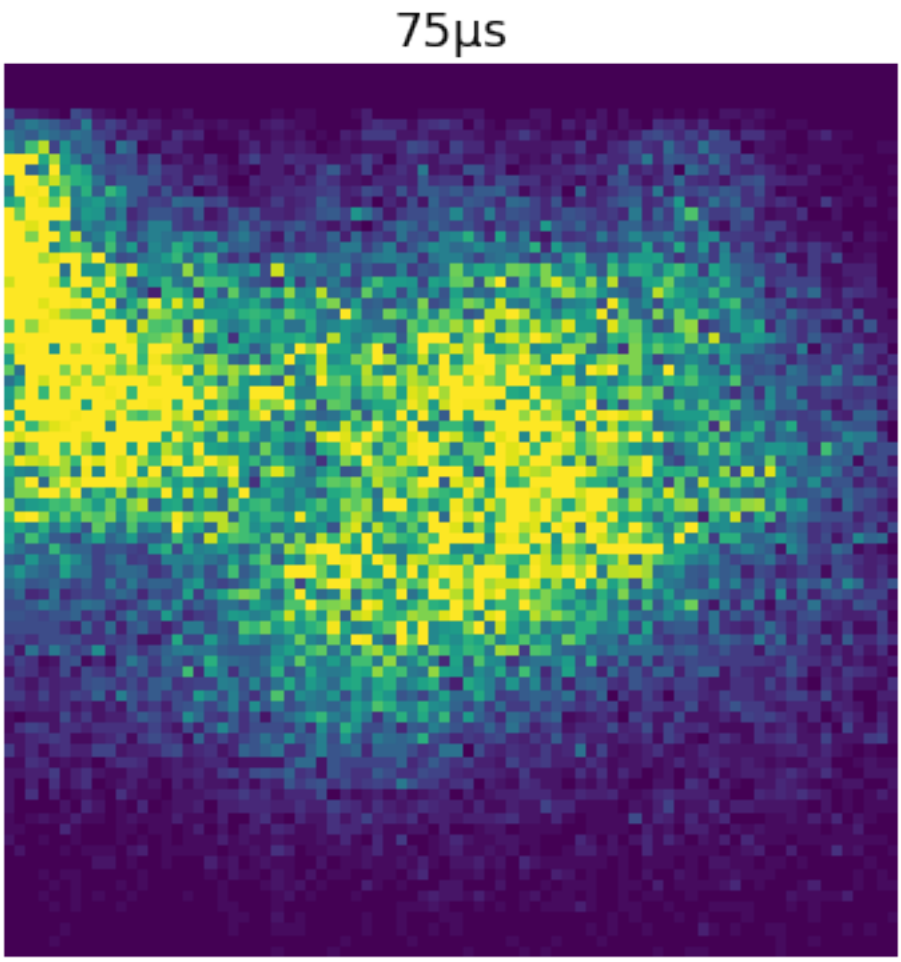

```
    Time=24.7  s
-1% Error
```



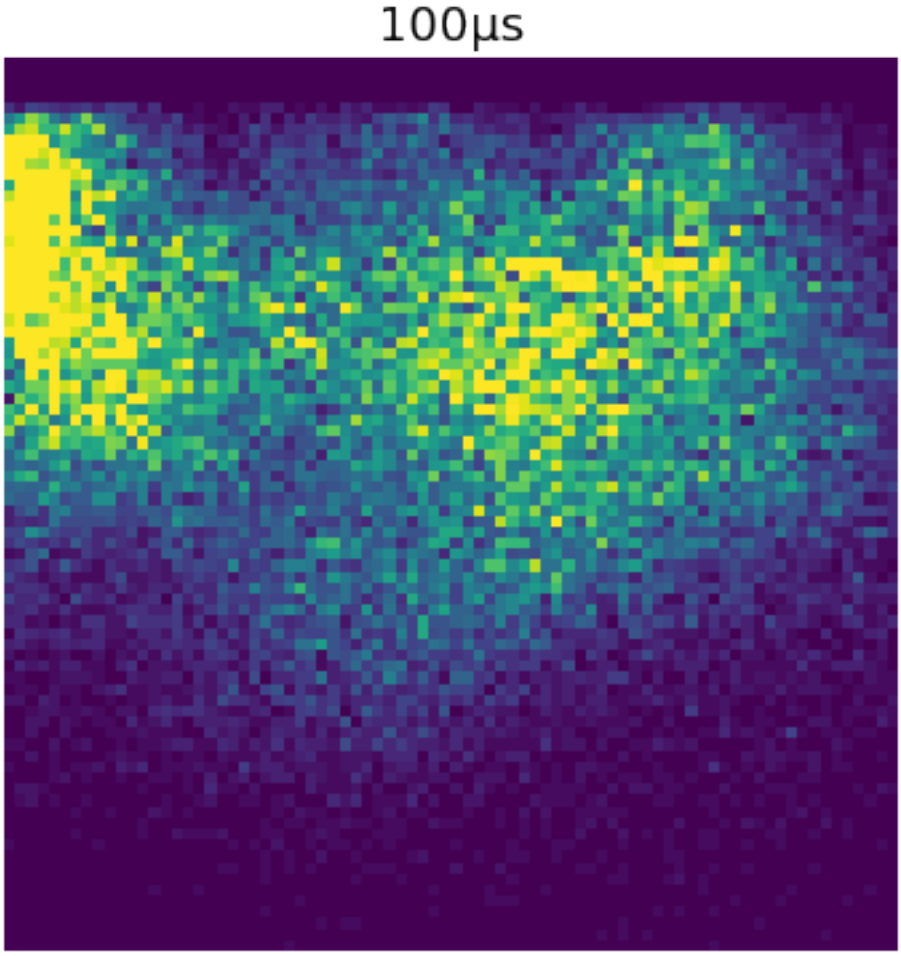
Time=50.3 s
1% Error



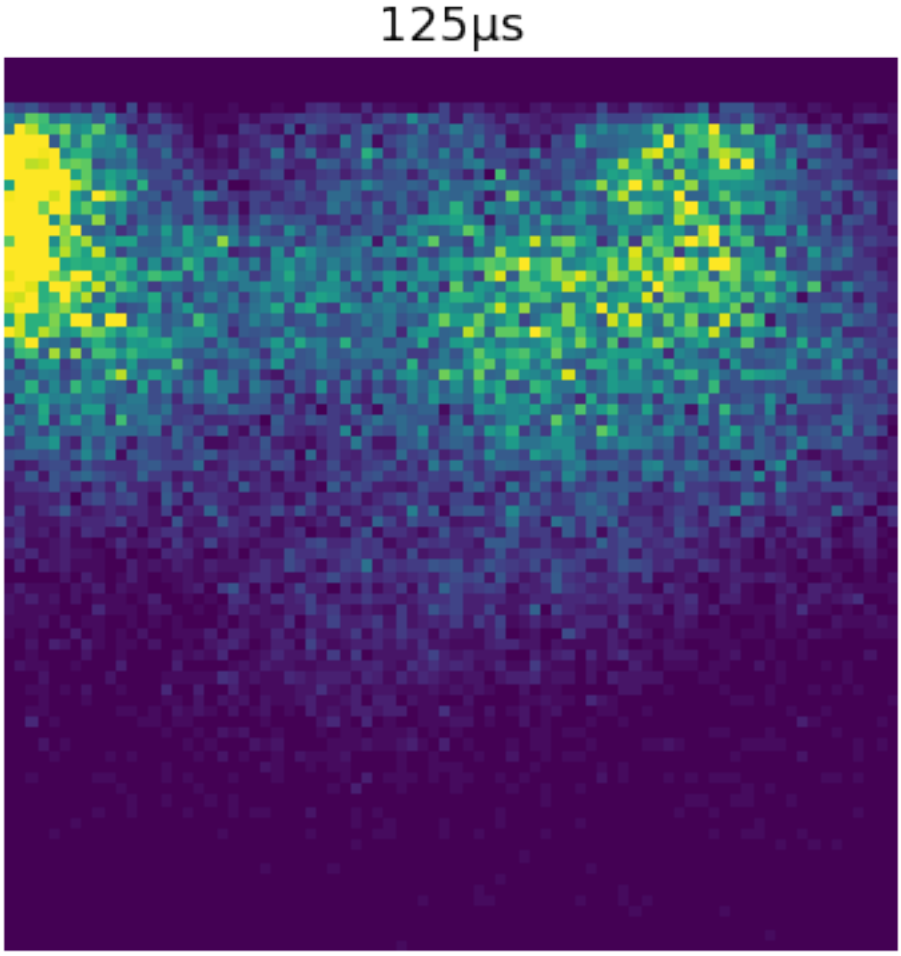

```
     Time=74.9  s
-0% Error
```



Time=99.0 s
-1% Error

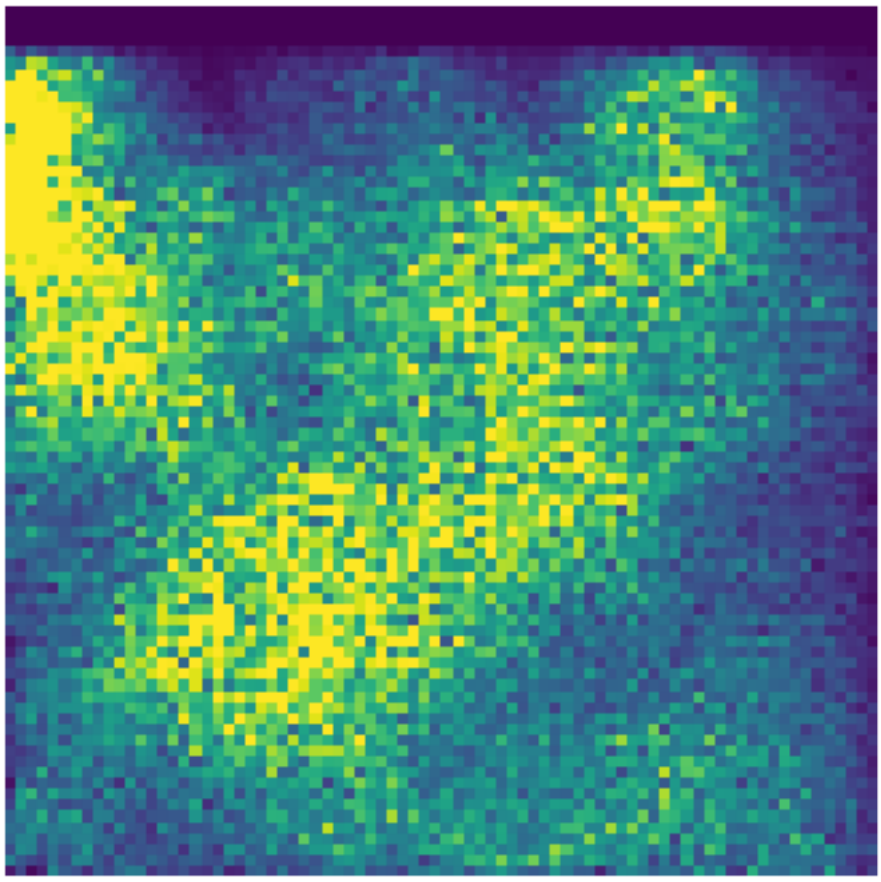


Time=124.9 s
-0% Error



[19]:
```
for time, image in zip(times, images):
    image.savefig("../figures/Time_of_Flight_%.0f_Microseconds.png" %␣
↪(time*10**6), dpi=300, bbox_inches='tight')
```

[20]:
```
time_of_flight_total = e.plot_total_density(figsize=(6, 6),␣
↪normalize_by_mean=True, colorbar=False, title=False, vmax=2)  # 5x brighter␣
↪than individual time images
time_of_flight_total.savefig("../figures/Time_of_Flight_Total.png", dpi=300,␣
↪bbox_inches='tight')
```



8 Figure 5h

[21]:
```
imp.reload(assets)
file_manager = assets.FileManager('../data/transport_simulations/')
# file_manager.add_files('ct_output_09_n100/', angle=0, field=10**6) original␣
 ↪version
file_manager.add_files('ct_28_n1000/', angle=90)
```

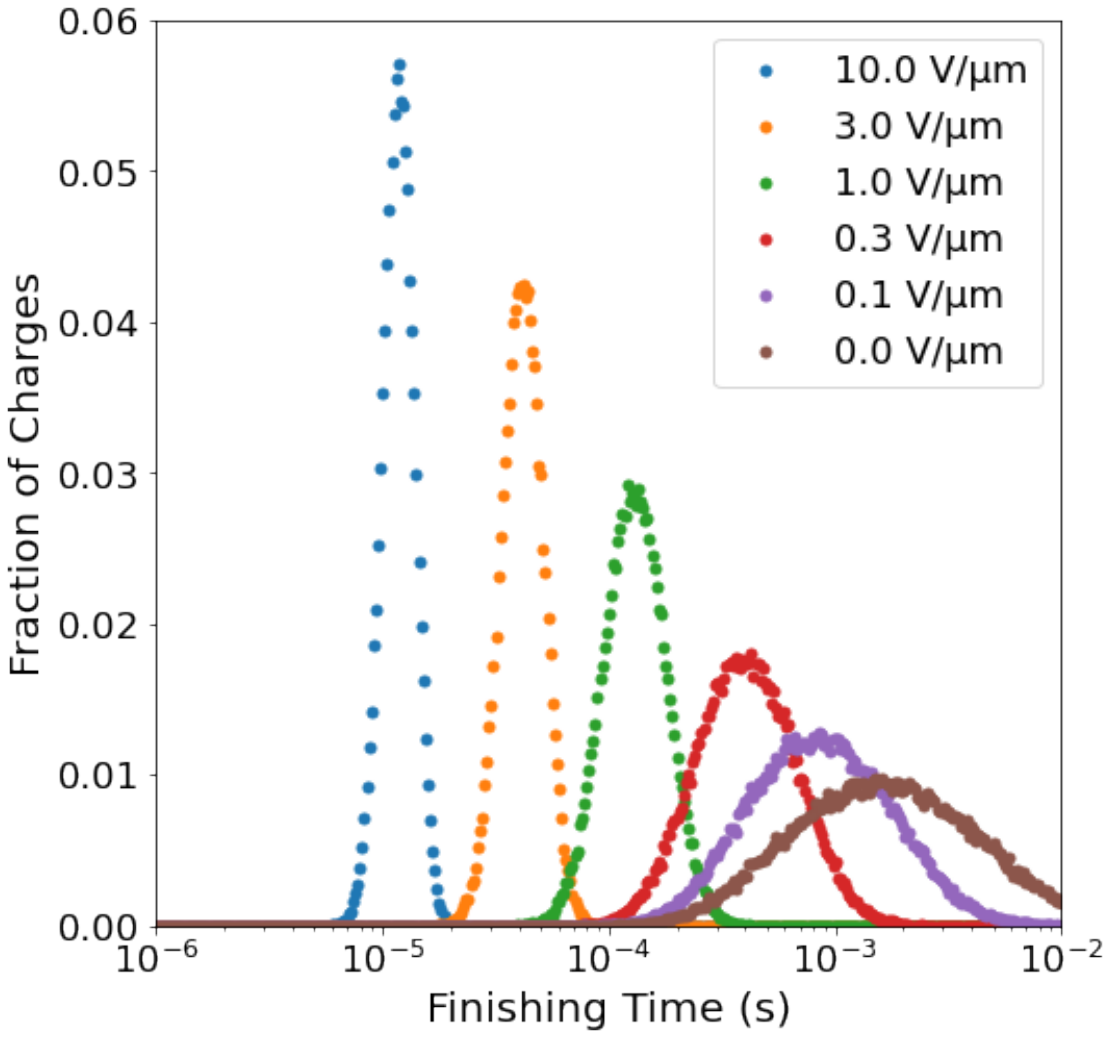
```
File Added: 1 Total Files
File Added: 2 Total Files
File Added: 3 Total Files
File Added: 4 Total Files
File Added: 5 Total Files
File Added: 6 Total Files
RAM memory % used: 63.4
RAM memory Available: 36.9 GB
```

[22]:
```
plt.figure(figsize=(8, 8))
for f in file_manager.files:
    f.unpack_results()
    change_in_active_fraction = np.zeros(len(f.time_seconds))
    change_in_active_fraction[1:] = -np.diff(f.active_fraction)
    plt.plot(f.time_seconds, change_in_active_fraction, marker='.',␣
 ↪markersize=10, linewidth=0, label='%.1f V/\u03bcm' % (f.field / 10**6))
    print('Mode: %.1f us' % (f.time_seconds[np.
 ↪argmax(change_in_active_fraction)] * 10**6) )
    print(f.field, f.grand_avg_velocity/10**6)
plt.xscale('log')
plt.xlim(10**-6, 10**-2)
plt.ylim(0)
plt.xlabel('Finishing Time (s)')
plt.ylabel('Fraction of Charges')
plt.legend()
plt.gca().set_box_aspect(1)
time_of_flight = plt.gcf()
plt.show()
```

```
/home2/luke/simulations_paper/src/charge_transfer_assets.py:1259:
RuntimeWarning: invalid value encountered in true_divide
  self.avg_velocity = np.mean(charge_velocity, axis=1)[:end] /
self.active_fraction[:end]  # Average at each time step

Mode: 11.7 us
10000000 66.72593260881236

/home2/luke/simulations_paper/src/charge_transfer_assets.py:1259:
RuntimeWarning: invalid value encountered in true_divide
  self.avg_velocity = np.mean(charge_velocity, axis=1)[:end] /
self.active_fraction[:end]  # Average at each time step

Mode: 41.7 us
3000000 19.154289821447822

/home2/luke/simulations_paper/src/charge_transfer_assets.py:1259:
RuntimeWarning: invalid value encountered in true_divide
  self.avg_velocity = np.mean(charge_velocity, axis=1)[:end] /
self.active_fraction[:end]  # Average at each time step

Mode: 120.2 us
1000000 6.288380561508398

/home2/luke/simulations_paper/src/charge_transfer_assets.py:1259:
RuntimeWarning: invalid value encountered in true_divide
  self.avg_velocity = np.mean(charge_velocity, axis=1)[:end] /
self.active_fraction[:end]  # Average at each time step

Mode: 426.6 us
300000 2.193199117559118
```

```
/home2/luke/simulations_paper/src/charge_transfer_assets.py:1259:
RuntimeWarning: invalid value encountered in true_divide
  self.avg_velocity = np.mean(charge_velocity, axis=1)[:end] /
self.active_fraction[:end]  # Average at each time step
```

```
Mode: 851.1 us
100000 1.1546744345060789
Mode: 1513.6 us
1e-10 0.6960003704861615
```



[23]:
```
time_of_flight.savefig('../figures/Time_of_Flight.png', dpi=300)
```

9 Figure 6a

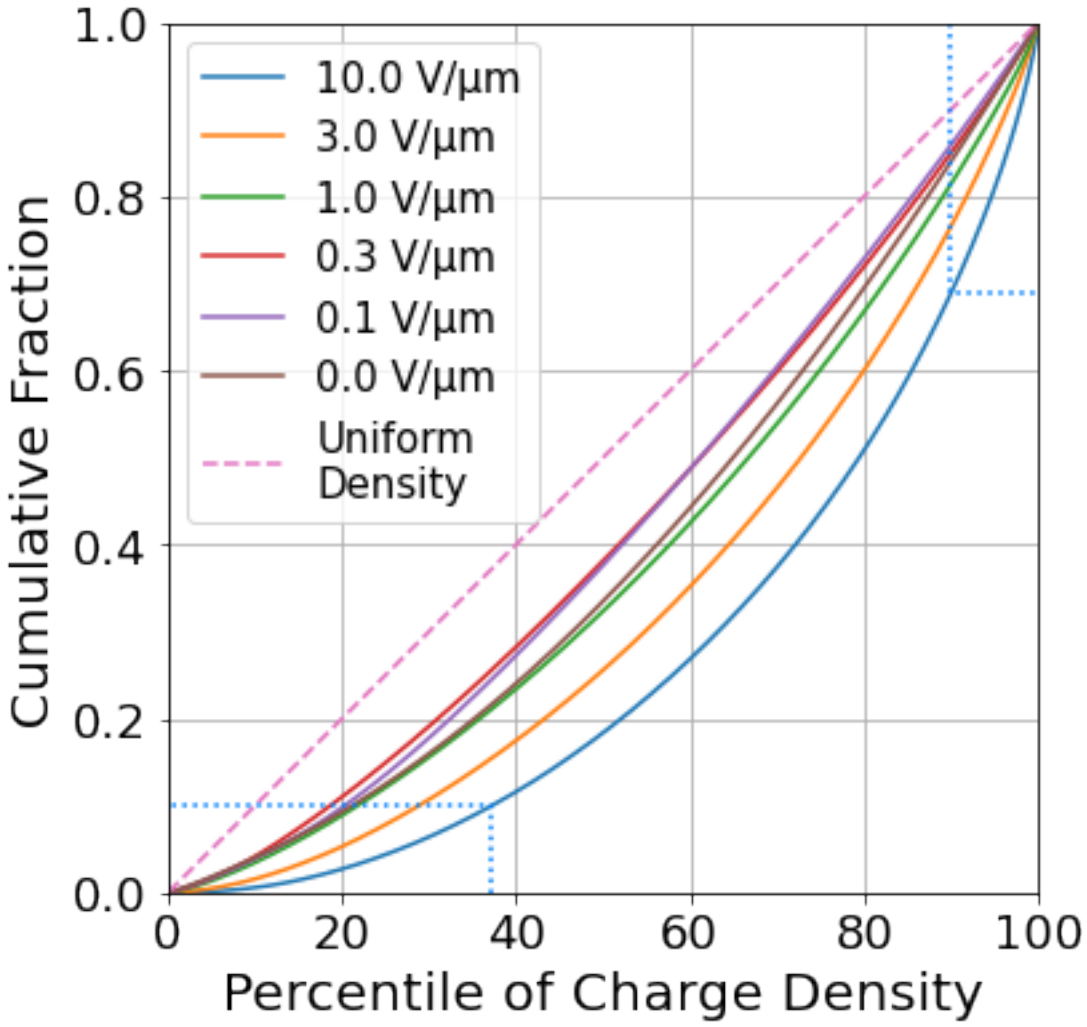
[24]:
```
imp.reload(assets)
file_manager = assets.FileManager('../data/transport_simulations/')
file_manager.add_files('ct_28_n1000/', angle=90, sign=1)
```

```
File Added: 1 Total Files
File Added: 2 Total Files
File Added: 3 Total Files
File Added: 4 Total Files
File Added: 5 Total Files
File Added: 6 Total Files
RAM memory % used: 63.7
RAM memory Available: 36.7 GB
```

[25]:
```
plt.figure(figsize=(6, 6))
for f in file_manager.files:
    lorenz_curve, gini = f.get_lorenz_curve()
    x = np.arange(len(lorenz_curve))  / len(lorenz_curve) * 100  # Number of␣
 ↪bins is less than 6400 due to wide absorbing boundary.
    plt.plot(x, lorenz_curve, label='%.1f V/\u03bcm' % (f.field / 10**6))

    if f.field == 10**7:
        x_sample = np.array(x)
        lorenz_sample = np.array(lorenz_curve)
plt.plot(x, x/100, '--', label='Uniform\nDensity')
plt.xlim(0, 100)
plt.ylim(0, 1)
plt.gca().set_box_aspect(1)
plt.grid()
plt.legend(fontsize='small')
plt.xlabel('Percentile of Charge Density')
plt.ylabel('Cumulative Fraction')

point_1 = np.argmin((x_sample-90)**2)
x_1 = x_sample[point_1]
y_1 = lorenz_sample[point_1]
plt.plot([x_1, x_1, 100], [1, y_1, y_1], ':', c=(0, 0.5, 1.0))
print('The top 10%% of bins have %.0f%% of the charge density' % ((1 - y_1) *␣
 ↪100))

point_2 = np.argmin((lorenz_sample-0.1)**2)
x_2 = x_sample[point_2]
y_2 = lorenz_sample[point_2]
plt.plot([x_2, x_2, 0], [0, y_2, y_2], ':', c=(0, 0.5, 1.0))
print('The bottom %.0f%% of bins have 10%% of the charge density' % x_2)
```

```
lorenz_figure = plt.gcf()
plt.show()
```

```
The top 10% of bins have 31% of the charge density
The bottom 37% of bins have 10% of the charge density
```

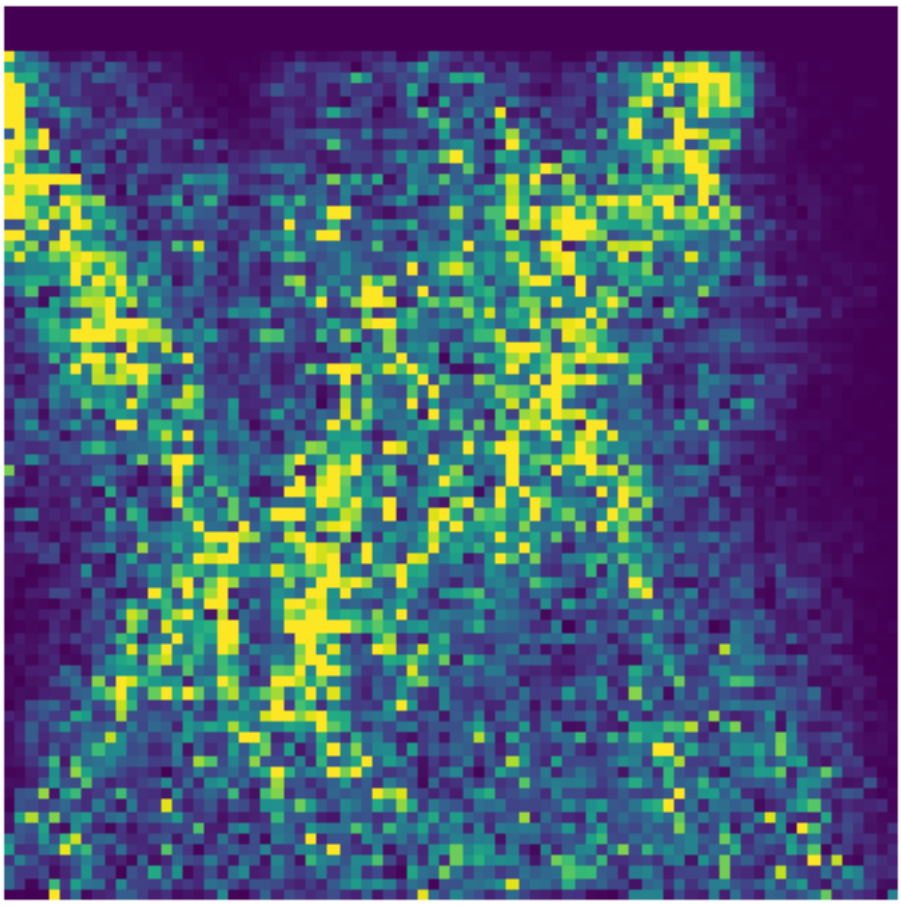


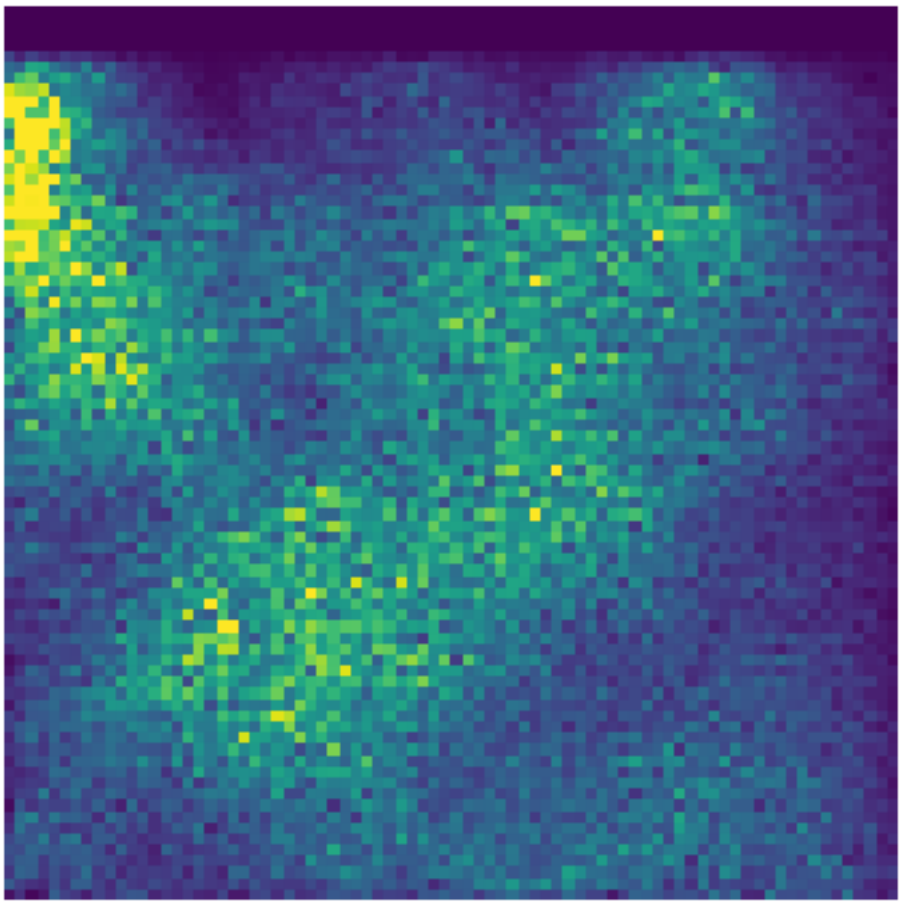
[26]:
```
lorenz_figure.savefig('../figures/Lorenz_Figure.png', dpi=300)
```

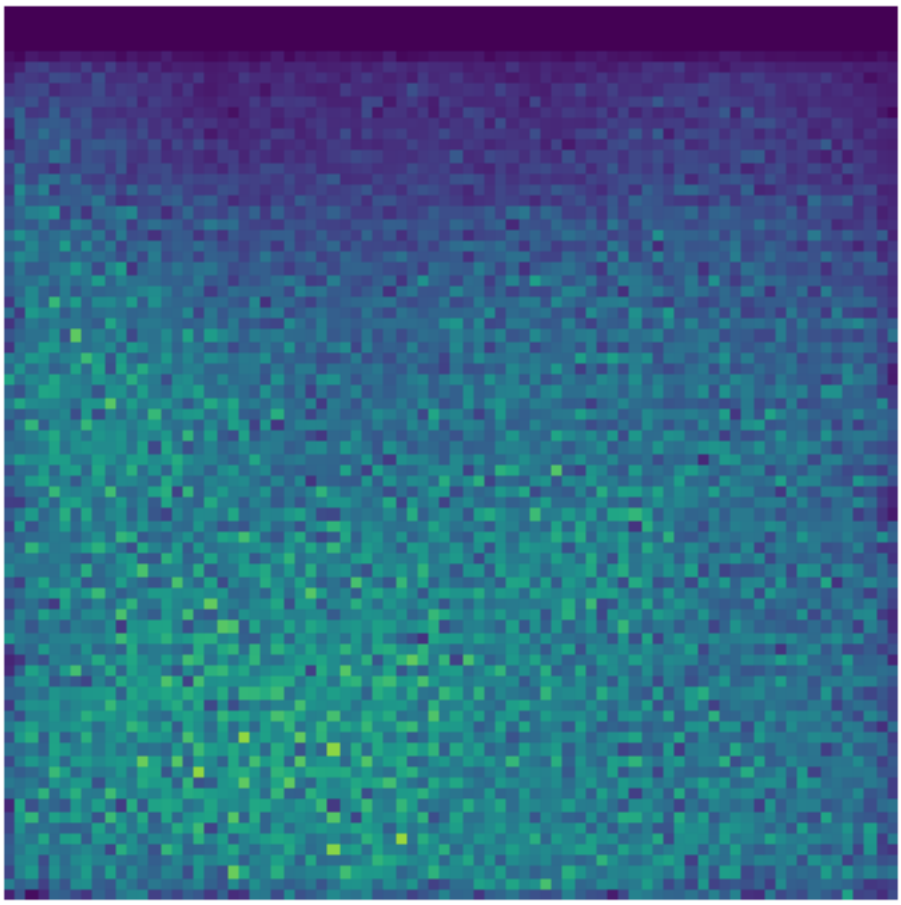
[27]:
```
for f in file_manager.files:
    if f.field in [10**5, 10**6, 10**7]:
        f.unpack_results()
        vmax=3
        new = f.plot_total_density(figsize=(6, 8), normalize_by_mean=True,␣
↪vmax=vmax, colorbar=False, title=False)
```

```
        new.savefig('../figures/Density_Variation_%.1fV_per_um.png' % (f.field/
↪10**6), dpi=300, bbox_inches='tight')
```

```
/home2/luke/simulations_paper/src/charge_transfer_assets.py:1259:
RuntimeWarning: invalid value encountered in true_divide
  self.avg_velocity = np.mean(charge_velocity, axis=1)[:end] /
self.active_fraction[:end]  # Average at each time step
```







10 Figure 7

[28]:
```
imp.reload(assets)
file_manager = assets.FileManager('../data/transport_simulations/')
file_manager.add_files('ct_16_n100/', angle=0, sign=1, field=10**6)
```

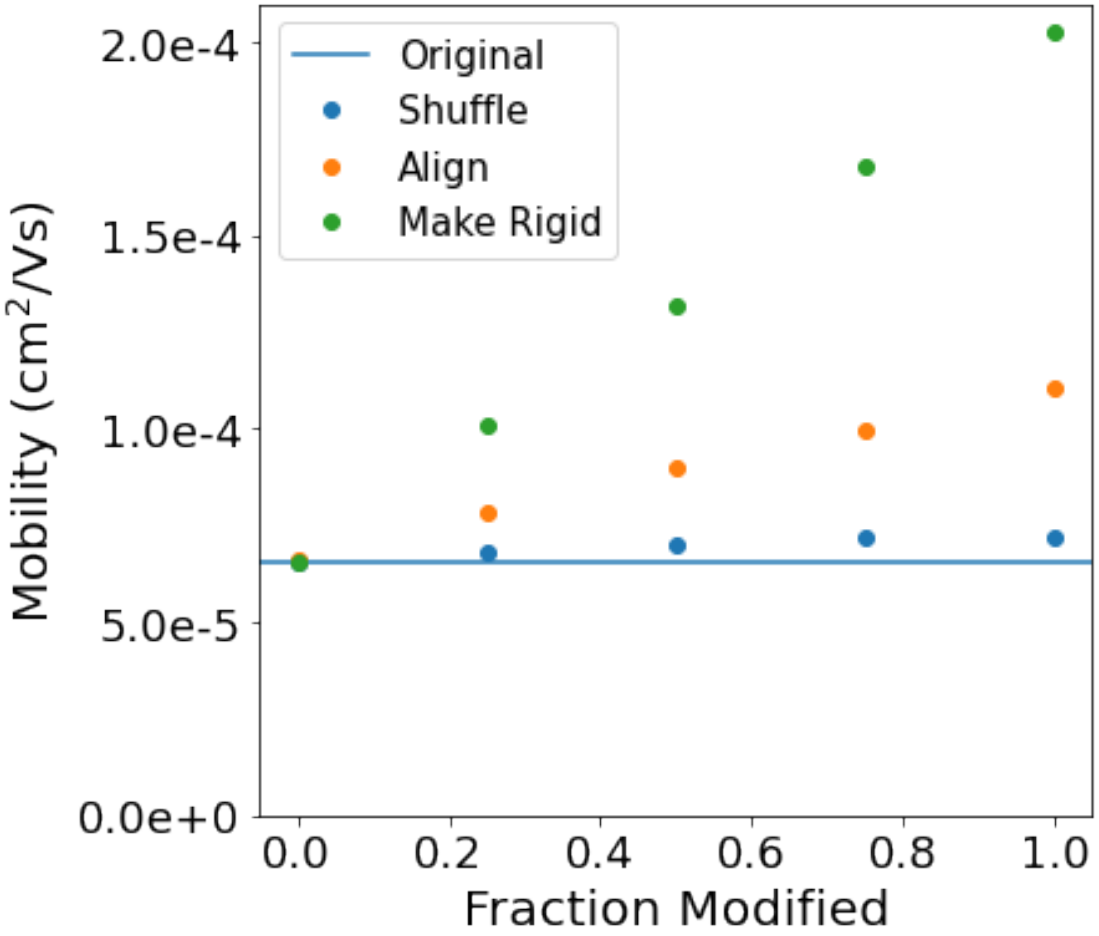
```
File Added: 1 Total Files
File Added: 2 Total Files
File Added: 3 Total Files
File Added: 4 Total Files
File Added: 5 Total Files
File Added: 6 Total Files
File Added: 7 Total Files
File Added: 8 Total Files
File Added: 9 Total Files
File Added: 10 Total Files
```

```
File Added: 11 Total Files
File Added: 12 Total Files
File Added: 13 Total Files
File Added: 14 Total Files
File Added: 15 Total Files
RAM memory % used: 66.4
RAM memory Available: 33.9 GB
```

[29]:
```
def remove_character(string, position):
    return string[:position] + string[1 + position:]

shuffle = file_manager.get_files_where(shuffle=True, align=False,␣
 ↪make_rigid=False)
align = file_manager.get_files_where(shuffle=False, align=True,␣
 ↪make_rigid=False)
make_rigid = file_manager.get_files_where(shuffle=False, align=False,␣
 ↪make_rigid=True)

modify_fraction = np.linspace(0, 1, 5, endpoint=True)

plt.figure(figsize=(6, 6))
base_mobility = shuffle[0].get_plateau_mobility()
plt.axhline(y=base_mobility)
for dataset in [shuffle, align, make_rigid]:
    for e in dataset:
        e.unpack_results()
    n = len(dataset)
    mobilities = [e.get_plateau_mobility() for e in dataset]  # cm2/Vs
    velocities = np.array(mobilities) * 10**4  # cm/s
    # distance_per_hop = velocities / interchain_rate * 10**7  # nm
    plt.plot(modify_fraction[:n], mobilities[:n], marker='.', markersize=12,␣
 ↪linewidth=0)


plt.ylim(0, 0.00021)
plt.xlabel('Fraction Modified')
plt.ylabel('Mobility (cm$^2$/Vs)')

ax1 = plt.gca()
y_limits = ax1.get_ylim()
plt.yticks(np.linspace(0, 2*10**-4, 5))
ax1.set_yticklabels([remove_character('%.1e' % x, -2) for x in ax1.get_yticks().
 ↪tolist()])

ax1.legend(['Original', 'Shuffle', 'Align', 'Make Rigid'], fontsize='small')
modifications_figure = plt.gcf()
plt.gcf().savefig('../figures/Modifications.png', bbox_inches='tight', dpi=300)
```

```
plt.show()
```

```
/home2/luke/simulations_paper/src/charge_transfer_assets.py:1259:
RuntimeWarning: invalid value encountered in true_divide
  self.avg_velocity = np.mean(charge_velocity, axis=1)[:end] /
self.active_fraction[:end]  # Average at each time step
/home2/luke/simulations_paper/src/charge_transfer_assets.py:1259:
RuntimeWarning: invalid value encountered in true_divide
  self.avg_velocity = np.mean(charge_velocity, axis=1)[:end] /
self.active_fraction[:end]  # Average at each time step
```



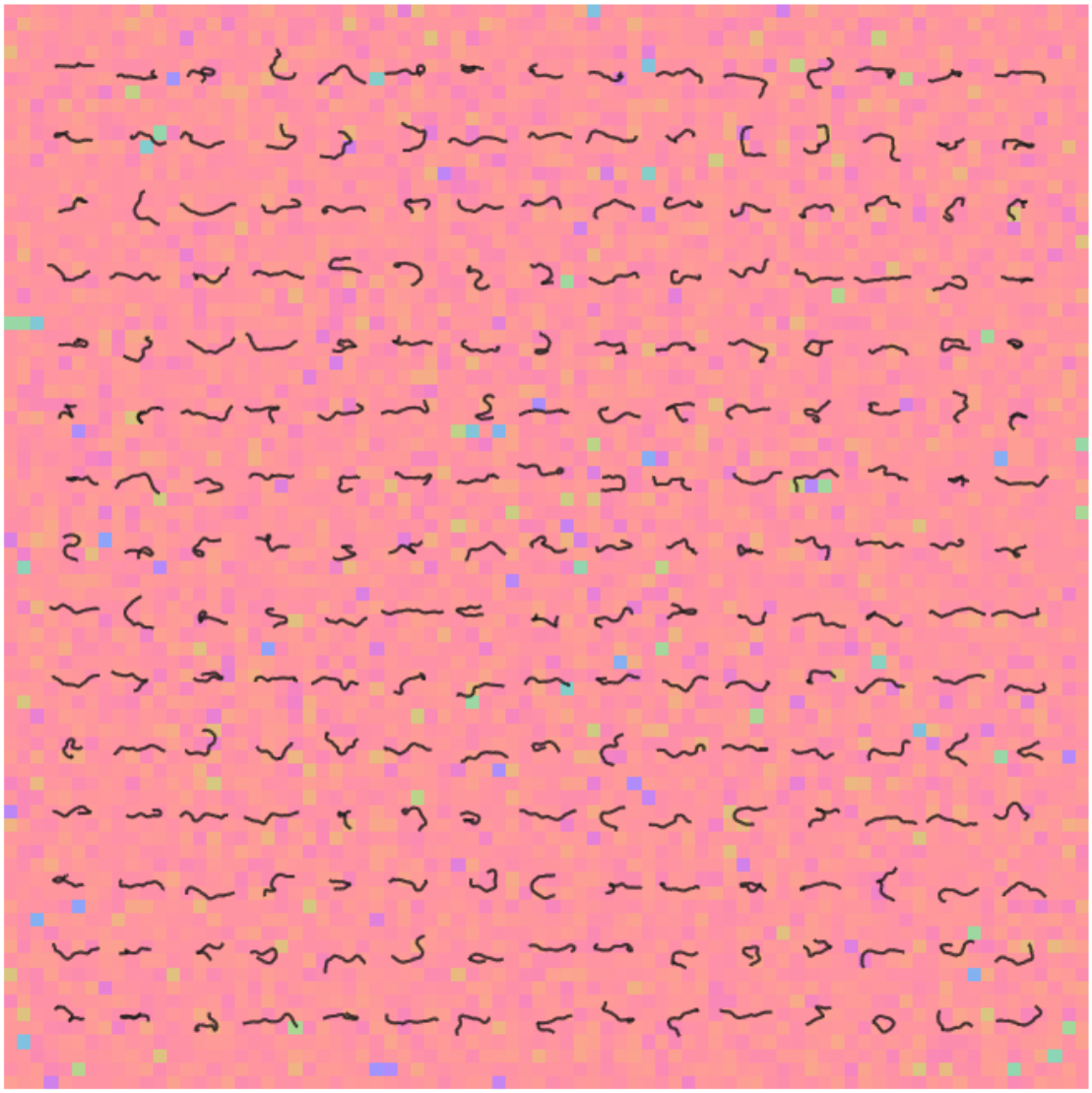
[30]:
```
file_manager = assets.FileManager('../data/transport_simulations/')
file_manager.add_files('ct_16_n100/', angle=0, sign=1, field=10**6,␣
 ↪modify_fraction=1.0)
```

```
File Added: 1 Total Files
File Added: 2 Total Files
File Added: 3 Total Files
```

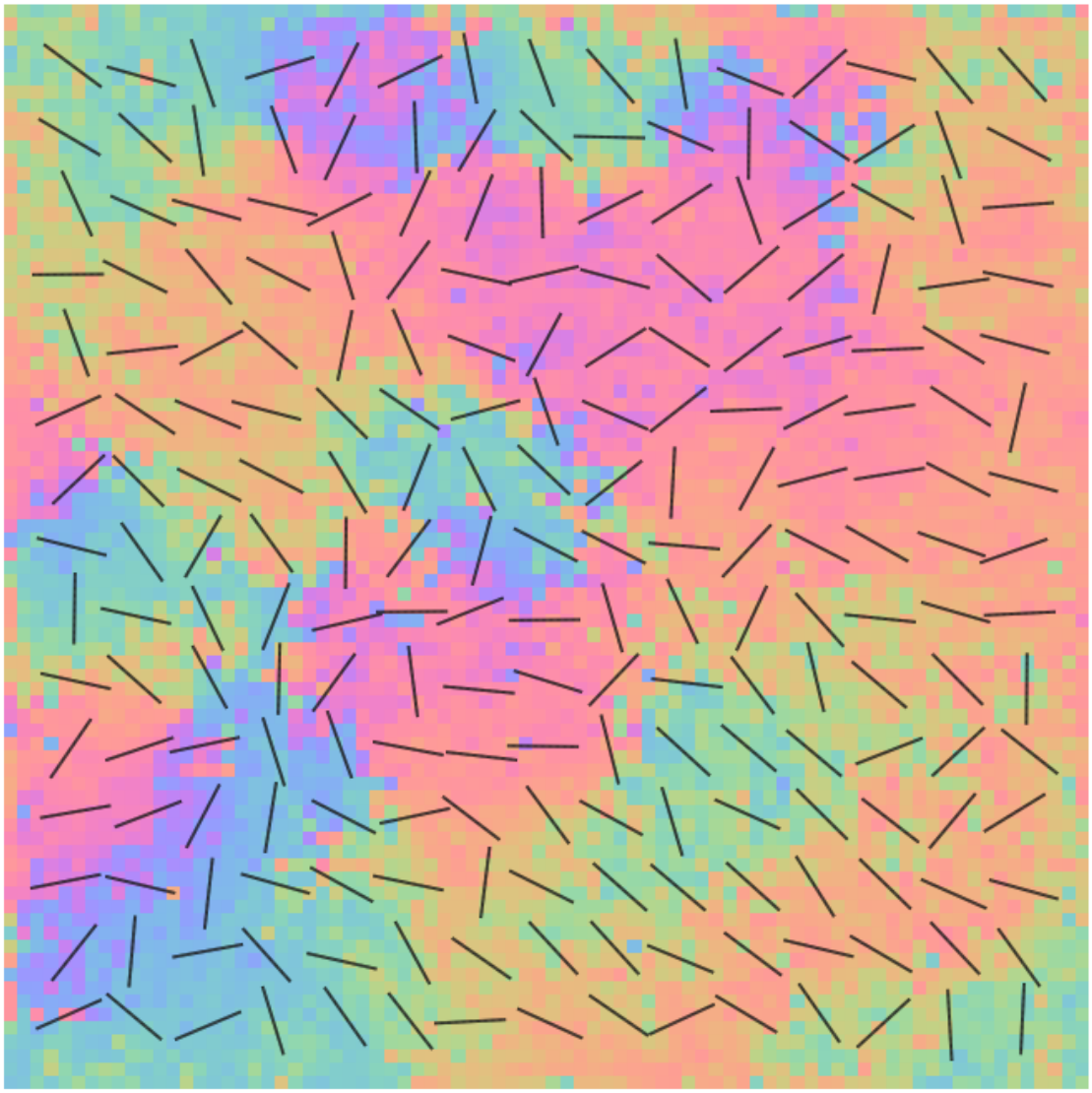
```
RAM memory % used: 67.1
RAM memory Available: 33.2 GB
```

[31]:
```
for f in file_manager.files:
    plt.figure(figsize=(6, 6))
    chains = f.chains
    chains_image = chains.plot_lines(grid=(15, 15), linewidth=2, method='Color␣
 ↪by Bin', alpha=0.5)
    if chains.shuffle:
        name = 'chains_shuffled'
    elif chains.make_rigid:
        name = 'chains_rigid'
    elif chains.align:
        name = 'chains_aligned'
    chains_image.savefig('../figures/%s.png' % name, dpi=300,␣
 ↪bbox_inches='tight')
    plt.show()
```

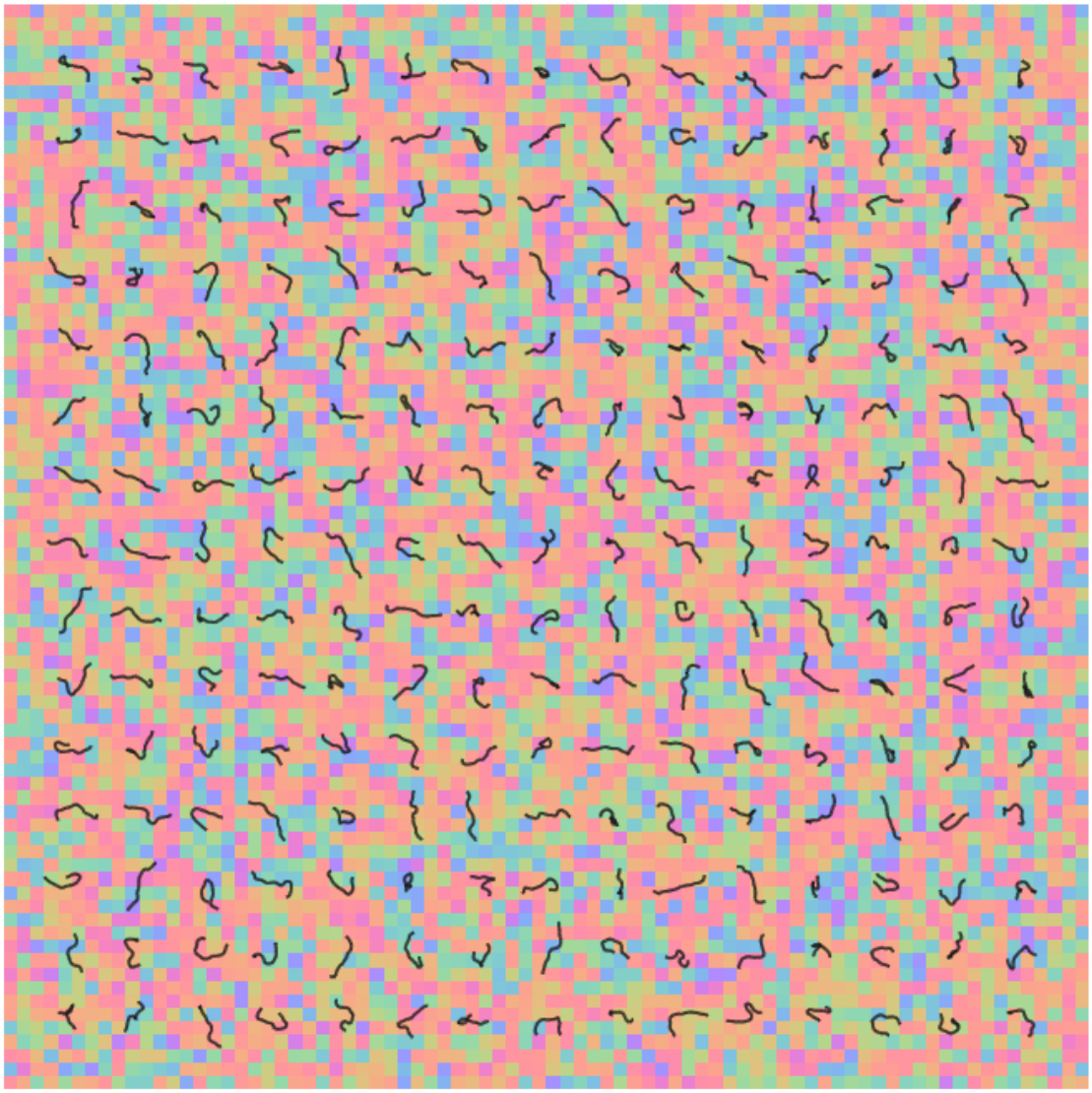
```
<Figure size 432x432 with 0 Axes>
```



<Figure size 432x432 with 0 Axes>



```
<Figure size 432x432 with 0 Axes>
```



[ ]: