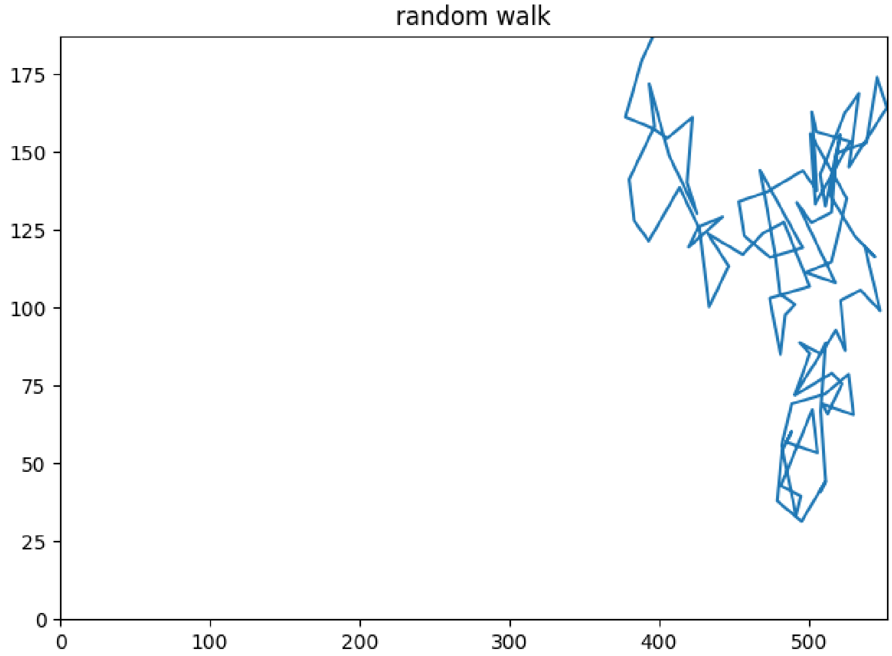
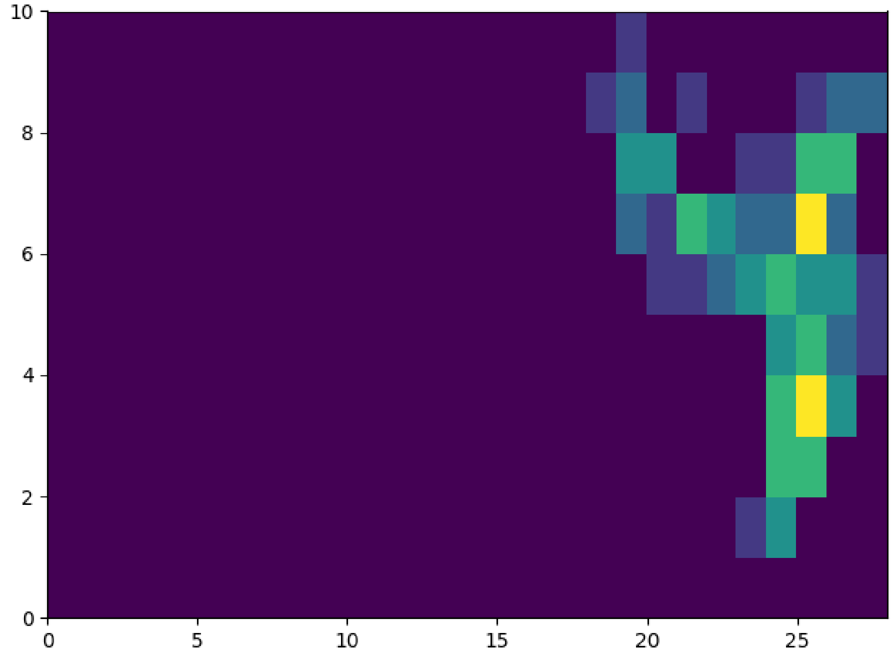
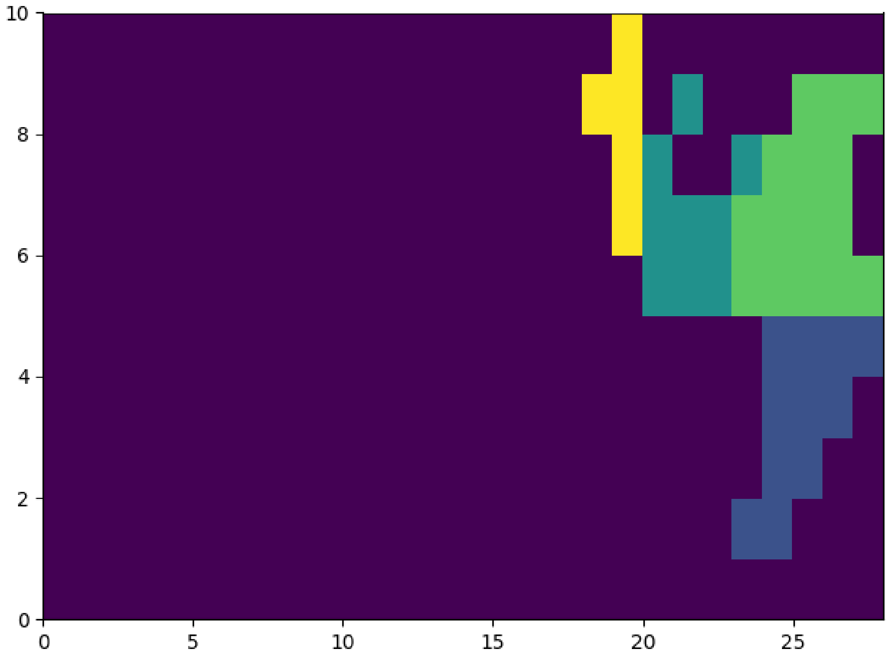
The random walk:



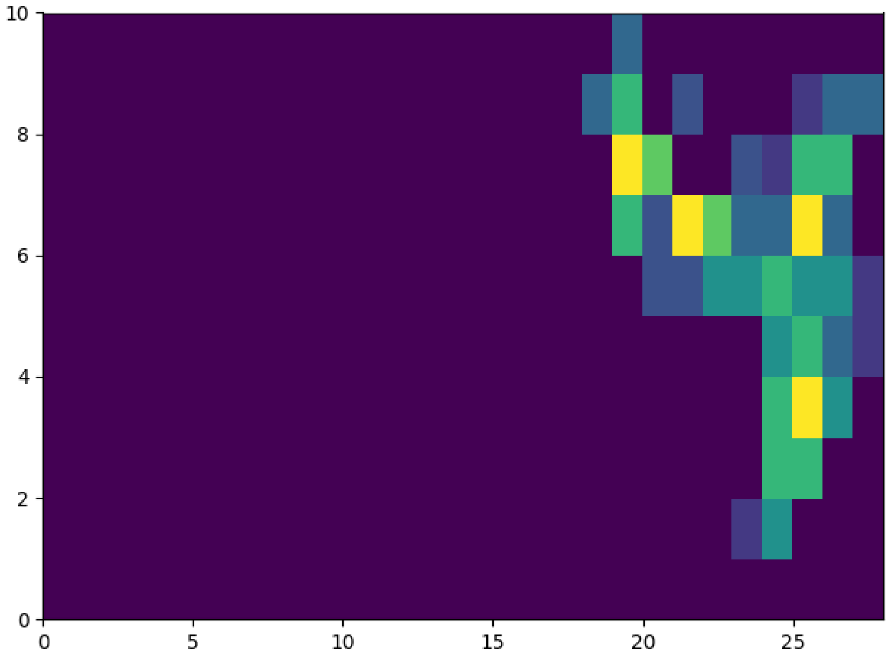
Counts on grid:



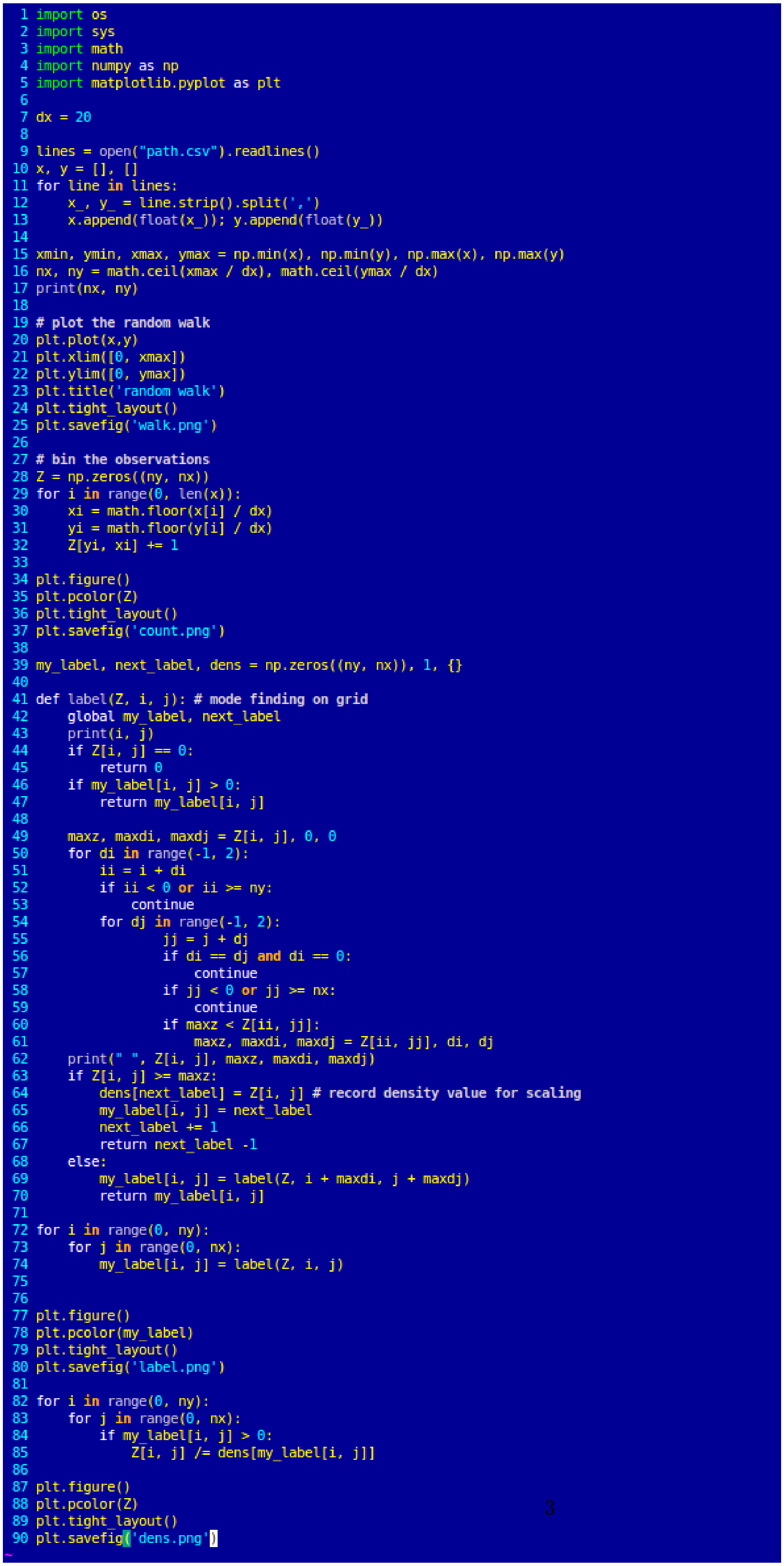
Mode label:



Count (density) normalized into [0, 1] for each cluster:



Source code



```python
import os
import sys
import math
import numpy as np
import matplotlib.pyplot as plt

dx = 20

lines = open("path.csv").readlines()
x, y = [], []
for line in lines:
    x_, y_ = line.strip().split(',')
    x.append(float(x_)); y.append(float(y_))

xmin, ymin, xmax, ymax = np.min(x), np.min(y), np.max(x), np.max(y)
nx, ny = math.ceil(xmax / dx), math.ceil(ymax / dx)
print(nx, ny)

# plot the random walk
plt.plot(x,y)
plt.xlim([0, xmax])
plt.ylim([0, ymax])
plt.title('random walk')
plt.tight_layout()
plt.savefig('walk.png')

# bin the observations
Z = np.zeros((ny, nx))
for i in range(0, len(x)):
    xi = math.floor(x[i] / dx)
    yi = math.floor(y[i] / dx)
    Z[yi, xi] += 1

plt.figure()
plt.pcolor(Z)
plt.tight_layout()
plt.savefig('count.png')

my_label, next_label, dens = np.zeros((ny, nx)), 1, {}

def label(Z, i, j): # mode finding on grid
    global my_label, next_label
    print(i, j)
    if Z[i, j] == 0:
        return 0
    if my_label[i, j] > 0:
        return my_label[i, j]

    maxz, maxdi, maxdj = Z[i, j], 0, 0
    for di in range(-1, 2):
        ii = i + di
        if ii < 0 or ii >= ny:
            continue
        for dj in range(-1, 2):
                jj = j + dj
                if di == dj and di == 0:
                    continue
                if jj < 0 or jj >= nx:
                    continue
                if maxz < Z[ii, jj]:
                    maxz, maxdi, maxdj = Z[ii, jj], di, dj
    print(" ", Z[i, j], maxz, maxdi, maxdj)
    if Z[i, j] >= maxz:
        dens[next_label] = Z[i, j] # record density value for scaling
        my_label[i, j] = next_label
        next_label += 1
        return next_label -1
    else:
        my_label[i, j] = label(Z, i + maxdi, j + maxdj)
        return my_label[i, j]

for i in range(0, ny):
    for j in range(0, nx):
        my_label[i, j] = label(Z, i, j)


plt.figure()
plt.pcolor(my_label)
plt.tight_layout()
plt.savefig('label.png')

for i in range(0, ny):
    for j in range(0, nx):
        if my_label[i, j] > 0:
            Z[i, j] /= dens[my_label[i, j]]

plt.figure()
plt.pcolor(Z)
plt.tight_layout()
plt.savefig('dens.png')
```