```
####################################################################################
#
# STKDE.py
# Author:   Griffen Angel/modified by E. Delmelle (delmelle@gmail.com), Meijuan Jia,
Coline Dony
# Updates: February 24 2012 / March 23 2012 / April 12 2012 / April 16 2013
# Purpose: To write a code which takes point data input and creates a file to be
read
#               by the Voxler program, that illustrates a 3D (or Space-Time) Kernel
Density Estimation
# Input:      Text file containing X,Y, T coordinates
#               Spatial bandwidth parameter (h_s), temporal bandwidth parameter
(h_t) and voxel size
#               Spatial bandwidth parameter (h_s), temporal bandwidth parameter
(h_t) and voxel size
# Output:   Text file containing X,Y,T coordinates of the Voxel where STKDE is
carried out. Also
#               contains a Z-field, which is the intensity
####################################################################################
#
import math, datetime, sys, os

## Input Model Parameters
hs= 1000                 #spatial bandwidth (here in meters)
ht = 10                  # temporal bandwidth (here in days)
spaceWindow = 100  # width of space bin (here in meters)
timeWindow  = 1        # width of time bin (here in days)

# Location of the input txt file containing the data
Cases_inFile  = open("data.txt",'r')

# STKDE formula (see Nakaya & Yano, 2010 in Transactions in GIS)
# some formula specific parameters may change depending on the nature of the data,
shape of the kernel..
def densityF(x, y, t, xi, yi, ti, n, hs, ht):
    u = (x-xi) / hs
    v = (y-yi) / hs
    w = (t-ti) / ht
    if  pow(u, 2) + pow(v, 2) < 1 and pow(w, 2) < 1:
        constantTerm = pow(10.0, 10) / (n * pow(hs, 2) * ht)
        Ks = (2 / math.pi) * (1 - pow(u, 2) - pow(v, 2))
        Kt = 0.75 * (1 - pow(w, 2))
        spaceTimeKDE = constantTerm * Ks * Kt
    else: spaceTimeKDE = 0
    return spaceTimeKDE

# Starting time of the STKDE task
timeStart = datetime.datetime.now()

# loop to read point data and inserting it into distinct lists
X, Y, Z = zip(*[map(float, record.split()) for record in Cases_inFile.readlines()])
Cases_inFile.close()


# Defining the space-time extent of the task (use projected coordinate units)
minX =
maxX =
minY =
maxY =
minT =
maxT =
numberCases_total = len(X)
```

```
x_filtered, y_filtered, t_filtered = zip(*[record for record in zip(*[X, Y, Z]) if
record[0] >= minX - hs and record[0] <= maxX + hs])
del X, Y, Z

# Checking whether the point data falls within the portion to be computed by this
particular core
# If not within the portion assigned to that core, skip, else compute the STKDE
(reduction of computation time)
resultsSTKDE = []
startX = minX
while startX <= maxX:
    startY = minY
    while startY <= maxY:
            for startTime in range(int(minT), int(maxT), timeWindow):
                    xLeft, xRight = startX - hs, startX + hs
                    yBottom, yTop = startY - hs, startY + hs
                    tBottom, tTop = startTime - ht, startTime + ht
                    density = 0
                    for i in range(len(x_filtered)):
                            if x_filtered[i] >= xLeft and x_filtered[i] <= xRight
and y_filtered[i] >= yBottom and y_filtered[i] <= yTop and t_filtered[i] >= tBottom
and t_filtered[i] <= tTop:
                                    density += densityF(startX, startY, startTime,
x_filtered[i], y_filtered[i], t_filtered[i], numberCases_total, hs, ht)
                                    resultsSTKDE.append("%s\t%s\t%s\t%s\n" %
(startX, startY, startTime, density))
            startY += spaceWindow
    startX += spaceWindow
    print startX

outSuffix = "spaceBandw%s_timeBandw%s" % (hs, ht)
resultsSTKDE_filename = os.path.join("Out_%s.txt" % (outSuffix))
resultsSTKDE_outFile = open(resultsSTKDE_filename, 'w')
resultsSTKDE_outFile.writelines(resultsSTKDE)
resultsSTKDE_outFile.close()

# Closing the computation time and storing it into a text file
timeEnd = datetime.datetime.now()
timeResults_fileName = os.path.join("timeOut_%s.txt" % (outSuffix))
timeResults = open(timeResults_fileName, 'w')
timeResults.write("%s\t%s\t%s\n" % (timeStart, timeEnd, timeEnd - timeStart))
timeResults.close()
```