sustainability

Supplementary material

Study on a Quantitative Approach to Geometrical Configurations of Urban Space

Huimin Ji, Yunlong Peng and Wowo Ding *

School of Architecture and Urban Planning, Nanjing University, Nanjing 210093, China; jeehuimin@gmail.com
* Correspondence: dww@nju.edu.cn

Script S1. Python Script of Spatial Characteristic Indices

```
# -*- coding: utf-8 -*-

import pandas as pd
import numpy as np
from collections import OrderedDict


VIEW_THRESHOLD = 300
CONTI_THRESHOLD = 10

df = []
with open('B.txt') as f:
    for line in f:
        df += [float(line.strip('\n'))]

df = np.array(df, dtype=float)
df = df.reshape(-1, 360)

view_point_n = df.shape[0]

sta_dict = {'std_var': [], 'full_area': [], 'not_open_area': [],
            'inconti_n': [], 'open_region_n': [], 'open_angle_list': [],
            'tot_open_angle': [], 'perimeter': [],
            'convex': [], 'tri_perimeter': [], 'throughput': [],
            'peri_area_ratio': [], 'shape': []}

sta_dict = OrderedDict(sta_dict)

for i in range(0, view_point_n):
    data = df[i]
    sta_dict['shape'] += [np.mean(data) * 1.0 / np.max(data)]
    data_r = np.roll(data, 1)
    full_area = 0.5 * data * data_r * np.sin(np.pi / 180.0)
    full_area = full_area.sum()
    sta_dict['full_area'] += [full_area]

    data_n = data.shape[0]
```

```
inconti = [True if np.abs(data[(i - 1 + data_n) % data_n] - data[i]) > CONTI_THRESHOLD
           else False for i in range(0, data_n)]
inconti_n = np.sum(inconti)
sta_dict['inconti_n'] += [inconti_n]

abs_sum = np.sum(np.abs(data - data_r))
# sta_dict['abs_sum'] += [abs_sum]

perimeter = np.sum(data * np.pi / 180)
sta_dict['perimeter'] += [perimeter]
tri_perimeter = 0.0
for i in range(0, data.shape[0]):
    v1, v2 = data[i], data[i - 1]
    tri_perimeter += np.sqrt(v1 ** 2 + v2 ** 2
                             - 2 * v1 * v2 * np.cos(np.pi / 180))
sta_dict['tri_perimeter'] += [tri_perimeter]

inconti = np.array(inconti, dtype=bool)
inconti_p = np.argwhere(inconti)[:, 0]

open_area = []
inter_points = []
# no open region
if inconti_n < 2:
    sta_dict['open_region_n'] += [0]
    sta_dict['open_angle_list'] += [[]]
    sta_dict['tot_open_angle'] += [0]
    sta_dict['throughput'] += [0]
    diff = np.abs(data - data_r)
    sta_dict['convex'] += [diff.sum() / (diff.sum() + perimeter)]
    var = ((data - data.mean()) ** 2 / data.shape[0]).sum()
    std_var = np.sqrt(var)
    sta_dict['std_var'] += [std_var]
    sta_dict['not_open_area'] += [sta_dict['full_area'][-1]]
    sta_dict['peri_area_ratio'] += [tri_perimeter / full_area]
else:
    # scan open region
    for j in range(inconti_n):
        if j < inconti_n - 1:
            open_area += [(data[inconti_p[j]: inconti_p[j + 1]].mean(),
                           inconti_p[j + 1] - inconti_p[j])]
            inter_points += [(inconti_p[j], inconti_p[j + 1])]
        else:
            tmp = data[inconti_p[j]:].sum() + data[:inconti_p[0]].sum()
            tmp = tmp / (data[inconti_p[j]:].shape[0] +
                         data[:inconti_p[0]].shape[0])
            inter_points += [(inconti_p[j], inconti_p[0])]
            open_area += [(tmp, 360 - inconti_p[j] + inconti_p[0])]

    is_open = False
    open_region_n = 1
    open_angle = []
```

```
            convex = 0
            perimeter = 0
            tri_perimeter = 0
            not_open_area = 0
            not_open = [False] * data.shape[0]
            for j in range(0, len(open_area)):
                area, angle = open_area[j]
                if open_area[j][0] >= open_area[j - 1][0]:
                    if not is_open:
                        open_region_n += 1
                        open_angle += [angle]
                        is_open = True
                    else:
                        open_angle[-1] += angle
                else:
                    l, u = inter_points[j][0], inter_points[j][1]
                    if l > u:
                        u = u + data.shape[0]
                    for k in range(l + 1, u):
                        v1 = data[k % data.shape[0]]
                        v2 = data[(k - 1) % data.shape[0]]
                        diff = np.abs(v1 - v2)
                        convex += diff
                        perimeter += v1 * np.pi / 180
                        tri_perimeter += np.sqrt(v1 ** 2 + v2 ** 2
                                                 - 2 * v1 * v2 * np.cos(1.0 / 180 * np.pi))
                        not_open_area += 0.5 * v1 * v2 * np.sin(np.pi / 180)
                        not_open[k % data.shape[0]] = True
                    is_open = False

            sta_dict['convex'] += [convex / (convex + perimeter)]

            sta_dict['perimeter'][-1] = perimeter
            sta_dict['tri_perimeter'][-1] = tri_perimeter
            sta_dict['not_open_area'] += [not_open_area]
            sta_dict['peri_area_ratio'] += [tri_perimeter / not_open_area]

            std_var = 0
            if np.array(not_open).sum() > 0:
                not_open_points = data[np.array(not_open)]
                var = ((not_open_points - not_open_points.mean()) ** 2
                       / not_open_points.shape[0]).sum()
                std_var = np.sqrt(var)
            sta_dict['std_var'] += [std_var]

            sta_dict['open_region_n'] += [open_region_n - 1]
            sta_dict['open_angle_list'] += [open_angle]
            sta_dict['tot_open_angle'] += [np.sum(open_angle)]
            sta_dict['throughput'] += [(open_region_n - 1) * np.sum(open_angle) / 360]

    output = pd.DataFrame(data=sta_dict, index=np.arange(1, view_point_n + 1))
```

```
    # output.drop(['abs_sum'], axis=1, inplace=True)
    print(output.columns)
    output.to_excel('Statistics.xls', header=[
            'Standard deviation', 'Total area', 'Part area', 'Point number', 'Open number', 'Open angel ', 'Open sum', 'Circumference', 'Convex', ' Triangle Perimeter', 'Open', 'Circularity', 'Shape'])
# output.to_excel('Statistics.xls', header=list(sta_dict.keys()))
```

Table S2. Statistics of the Spatial Characteristic Indices of Subspaces in Block A and B.

Number of Subspaces	Area [m²]	Shape [-]	Openness [-]
A0	2116	0.69	0.93
A1	2204	0.47	0.22
A2	1106	0.69	0.66
A3	2959	0.52	0.26
A4	1087	0.65	2.01
A5	6636	0.53	0.82
A6	477	0.24	0.08
A7	2432	0.50	0.28
A8	1173	0.51	0.51
A9	2197	0.60	0.31
A10	530	0.28	0.28
A11	913	0.53	1.10
A12	1516	0.43	0.63
A13	39	0.29	0.15
A14	143	0.60	0.53
A15	107	0.60	0.03
A16	98	0.40	0.43
A17	815	0.37	0.68
A18	96	0.27	0.15
A19	634	0.57	0.03
A20	122	0.69	0.48
A21	68	0.42	0.34
A22	48	0.40	0.29
A23	2879	0.25	0.26
A24	134	0.31	0.23
A25	100	0.79	0.71
A26	38	0.62	0.39
A27	25	0.52	0.24
A28	1394	0.19	0.08
A29	36	0.33	0.29
A30	27	0.64	0.43
A31	93	0.69	0.63
A32	927	0.50	0.64
A33	1253	0.45	0.32
A34	654	0.37	0.28
A35	361	0.46	0.55
B0	4080	0.69	0.47
B1	310	0.70	0.32
B2	2609	0.36	0.14

B3	1491	0.35	0.17
B4	3162	0.40	0.17
B5	319	0.18	0.03
B6	231	0.29	0.05
B7	1430	0.53	0.06
B8	933	0.46	0.30
B9	2311	0.42	0.23
B10	1010	0.64	0.24
B11	3804	0.69	0.63
B12	1278	0.42	0.13
B13	4288	0.64	0.40
B14	13394	0.73	1.11
B15	634	0.41	0.19
B16	3732	0.51	0.33

Table S3. Statistics of Spatial Average Wind Velocity Ratio of Subspaces in Block B.

Number of Spaces	Spatial Average Wind Velocity Ratio [-]
B0	1.08
B1	0.85
B2	1.19
B3	0.70
B4	1.69
B5	0.82
B6	0.69
B7	1.07
B8	1.24
B9	1.82
B10	0.71
B11	1.06
B12	0.59
B13	0.93
B14	1.10
B15	1.00
B16	1.86