Delineating Natural Cities from Large-Scale Images Using a Recursive Approach

Andy Jingqian Xue

LivableCityLAB, Urban Governance and Design Thrust, Society Hub
The Hong Kong University of Science and Technology (Guangzhou)
Email: andy.j.xue@connect.hkust-gz.edu.cn

Abstract
This tutorial delivers a step-by-step tutorial on identifying natural cities using global Nighttime Light (NTL) imagery. The NTL images are significant sources for measuring human activities and the progression of urbanization. By using the recursive approach based on the connected component analysis, we can not only uncover their inner hotspots of cities but also delineate the boundaries of natural cities on a global scale. The new recursive approach has significantly improved the efficiency of identifying natural cities. This tutorial begins by introducing natural cities and outlining the recursive generation process. After that, it elaborates the calculation and definition for the complexity of natural cities. Finally, the tutorial concludes with an interpretation of the temporal dynamics of complexity degree of the global urban spaces. The implementation of generating natural cities using Python 3 is available at https://github.com/AndyXue957/NaturalCitesfromGlobalNTL.

Keywords: Substructures, living structure, urban complexity analysis, scaling hierarchy of geographic space, global change

1. Introduction

Natural cities are human settlements that emerge organically and objectively from geographic big data analysis, in contrast to conventional cities, which are defined only by population metrics and organized in a top-down manner by governmental policies. Additionally, natural cities can be defined recursively while there are smaller but more important inner hotspots. This recursive notion has been widely recognized in many urban planning theories such as the central place theory (Christaller 1933, 1966) and fractal city (Mandelbrot 1984, Batty and Longley 1994). For example, the central place theory aims to determine the optimal spatial configuration of cities, organizing them based on size, number, and location, where each central city is recursively linked to six smaller cities, which in turn serve six even smaller cities. This recursive concept articulates the urban complexity organized by a scaling hierarchy where there are far more small cities than large ones (Jiang et al. 2015). The self-organized complexity reflects a non-linear and long-tailed statistical relationship between the scales and details of natural cities. Moreover, the complexity not only lies in space but can also be investigated in time. The Earth's surface is evolving and changing dynamically through the rapid development of technology and infrastructure.

The Nighttime Light (NTL) product is an effective tool for delineating natural cities, as it captures global and annual patterns of human activities through nighttime Earth observations. NTL imagery enables the identification of urban areas by measuring the intensity of nighttime brightness, which correlates with high levels of human concentration, whereas areas with lower brightness typically indicate lower activity levels in rural regions. In other words, bright pixels will be considered as urban pixels while dark pixels as rural pixels. Technically, natural cities are regarded as connected components of pixels, which is a highly computationally efficient representation of urban regions and make it possible for generating natural cities at global scale and reduce the time consumption within an acceptable range (see Appendix A for more details in the comparison).

In this tutorial, Section 2 introduces the concept of living structure and how complexity can be measured under the view of substructures. Then it provides the required data and Python scripts for delineating natural cities. Section 3 demonstrates the process of generating natural cities using global NTL imagery. Executed efficiently through a Python script, this approach allows for the rapid and automatic identification of natural cities across different iterations. The recursive process of generating natural

cities can be visualized in ArcGIS. Section 4 introduces the process of calculating the complexity of natural cities around the world. Eventually, Section 5 elaborates how to interpret the temporal dynamics of complexity and we can gain valuable insights into the developmental trends of urban spaces and the procedure of urbanization.

2. Living structure, urban complexity, and data for quantifying complexity

Living structure is the key concept of urban complexity and it is defined in a mathematical structure with hierarchy formed recursively by substructures. The Earth's surface is a typical living structure which is composed of natural cities. The discovery of living structure (Alexander 2002-2005) makes the complexity or goodness of space an objective fact which can be quantified mathematically, rather than a subjective opinion or feeling. For urban complexity analysis, the notion of substructures here is equivalent to the inner hotspots of natural cities. In terms of convenience, we refer natural cities to both themselves and their inner hotspots defined recursively. The degree of complexity (C) of urban spaces globally is determined by natural cities (NC) and their inherent scaling hierarchy (H). Simply put, the more natural cities it is and the higher levels of hierarchy of the inner hotspots in it, the higher the complexity of urban spaces. Therefore, the non-recursive complexity of the whole world can be defined as:

$$C = NC \times H \tag{1}$$

where NC and H denote the number of natural cities around the world, and their degree of inherent scaling hierarchy calculated by head/tail breaks (Jiang 2013), respectively. Natural cities can be derived more comprehensively by continuously decomposing the natural city until none of them are decomposable. All of the natural cities derived at different iterations constitute the living structure of the entire urban spaces across the globe. Therefore, the degree of urban complexity defined by the recursive approach (CR) is:

$$CR = \sum_{i=1}^{D} NC_i \times H_i \tag{2}$$

where D denotes the total number of decomposable natural cities and i is an individual decomposable natural city. The specific procedure of computing complexity for a global NTL imagery will be further illustrated in Section 4.

The two most prominent and widely used sources for NTL data are the Defense Meteorological Satellite Program (DMSP)/Operational Linescan System (OLS) and the Visible Infrared Imaging Radiometer Suite (VIIRS). However, the DMSP products are limited by their coarse spatial resolution (1,000m), whereas VIIRS data are only available from 2012 onwards. To identify natural cities across a broader temporal scope and with enhanced finer spatial resolution, we will employ a comprehensive annual global NTL dataset, synthesized by combining VIIRS and DMSP data (Chen et al. 2021), spanning from 2000 to 2022 with the spatial resolution of 500m. The annual dataset is open access and can be freely downloaded at: https://doi.org/10.7910/DVN/YGIVCD. Only the zip files for the years 2000 through 2022, specifically named as *2000_HasMask.zip*, *2001_HasMask.zip*, ..., *2022_HasMask.zip*, are required for this tutorial.

The recursive approach to delineating natural cities is programmed in Python 3 and the source code is available at https://github.com/AndyXue957/NaturalCitesfromGlobalNTL. The repository includes:

- *Data*: A folder to store all NTL images
    - *GD_2022.tif*: An example data for debugging the program on your own Python environment
- *Natural_cities*: A folder to store all generated natural cities as tif format file
- *Results_complexity*: A folder to store all statistics for the complexity degree of natural cities
- *generate_nc.py*: A Python script to recursively generate natural cities around the world
- *visualize_dynamics.py*: A Python script to interpret the temporal dynamics of natural cities
- *requirements.txt*: Requirements of python packages in this program

By utilizing the numpy, opencv-python and gdal packages, this implementation significantly outperforms the original ArcPy-based version in speed and memory efficiency, which enables users to delineate natural cities at global scale. The generated natural cities at different iterations will be saved in raster format, suitable for visualization in GIS applications like ArcGIS. Additionally, it will export the statistics of complexity to an Excel spreadsheet. The temporal dynamics of global complexity will be visualized through plots generated based on matplotlib package. Before you start running the python scripts, please ensure that you have installed all required Python packages by input *pip install -r requirements.txt* in command line.

3. Generating and visualizing natural cities

The natural cities can be extracted based on the brightness of NTL and generated as raster-based connected composed of bright pixels. The pixels darker than the average level of NTL value will be considered as rural areas reflecting lower levels of human activity, while the pixels lighter than the mean value will be considered as natural cities. However, the Earth's surface is composed of approximately 30% continents and 70% oceans, and NTL will be only detected on the land where people live in. Therefore, pixels representing the oceans has been set as NODATA in case it will significantly influence the average value of NTL. The entire process of generating natural cites based on a global NTL imagery will cost within 3 minutes which mainly depends on the hardware of your computer. You can follow the below steps to generate natural cities automatically:

1) Download an NTL imagery following Section 2 and put it under the folder *Data* in the program. Please rename it based on its sampling year, such as *2000.tif*, *2001.tif* etc (Panel b of Figure 1).

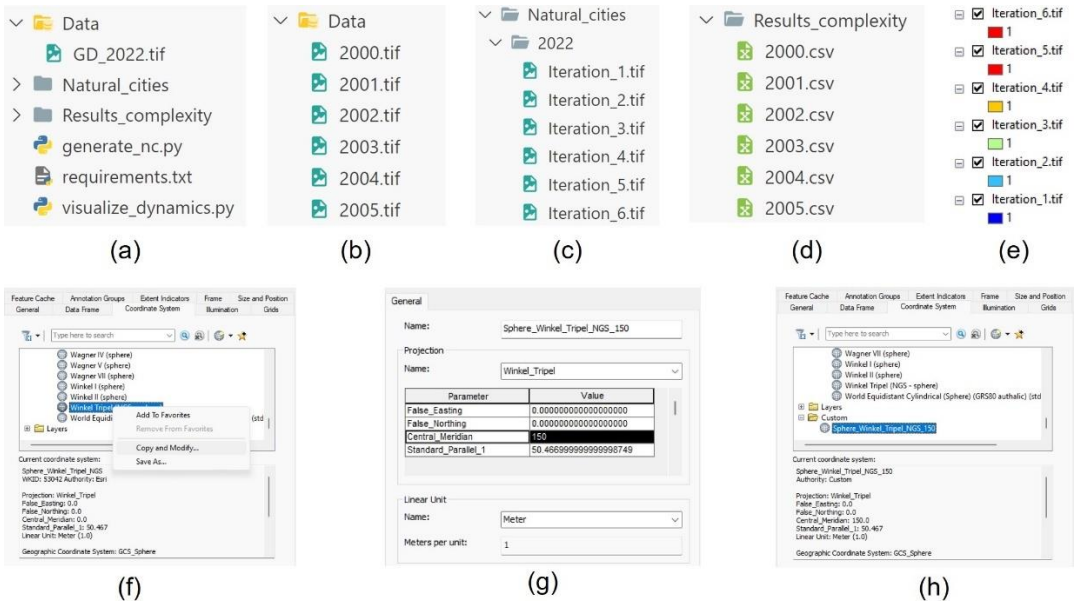

Figure 1: Key operations in generating natural cities

(Note: Panel a shows the initial file structure of the source code repository. Panel b to Panel d provide significant file settings or outputs during the generating process. Panel e elaborates the colour for natural cities generated in each iteration. Panel f to Panel h demonstrate the setting of projected coordinate system for global view of natural cities.)

2) Open python script *generate_nc.py* and adjust the parameters including *data_name* on line 57 based on the sampling year of the NTL imagery used, such as *2000*, *2001* etc.

3) Click *run* button on your IDE and the program will automatically generate the natural cities recursively, which the natural cities will be saved as tif format under *Natural_cities/%data_name%* folder, along with the calculated complexity saved as csv format under *Results_complexity*. The key function *cv2.connectedComponentsWithStats* to generate natural cities as connected

components is in *utils.py* on line 114. Please see official document at https://docs.opencv.org/5.x/ for more information.

4) Open the output folder *Natural_cities/%data_name%* and *Results_complexity* respectively to check whether the natural cities and their statistics results are successfully generated (Panel c and Panel d of Figure 1). Inside the *Natural_cities/%data_name%* folder, you should find I images, where I represents the total number of iterations. All natural cities generated at each iteration are saved as *Iteration_1.tif*, *Iteration_2, ... Iteration_I.tif*. In the *Results_complexity* folder, the complexity results for the NTL imagery will be stored in a file named *%data_name%.csv*.

You can follow the below steps to visualize the recursive process and s of generating natural cities, and visualize natural cities all around world:

1) Open ArcMap and import all generated natural cities under folder *Natural_cities/%data_name%* and arrange the input layers from largest iteration to smallest one (I to 1). For example, the layers in the table of contents in ArcMap should be ordered as *Iteration I*, *Iteration I-1*, …*Iteration_1* from top to bottom (Panel e of Figure 1).

2) Apply the *spectral* colormap ranging from pure blue to pure red in ArcGIS for the opened tif layers. Since each tif file of natural cities represents a binary image, you can only assign a single colour for the value of 1, with other pixels automatically set to transparent. The smallest iteration should be coloured in blue, with the colour gradually transitioning to red for the largest iteration. Because the number of natural cities generated in the last iteration might not be sufficient for clear visibility, it's recommended to combine and color the natural cities from iterations later than 5 in red to ensure they are visible (Panel e of Figure 1). Additionally, the natural cities are stored in several distinct files, thereby colours for each iteration cannot be obtained automatically. For the sake of convenience, we provide the RGB values of 5 classes from the first iteration to the iteration later than 5: (0,0,255), (51,194,255), (182,255,143), (255,200,0), (255,0,0).

3) Visualize the iteration process by selecting the purposed layers (Figure 2). For example, you can check *Iteration_1.tif* to visualize the natural cities, then check additionally *Iteration_2.tif* to visualize the natural cities generated at the second iteration, repeat it and do it again until the last iteration.

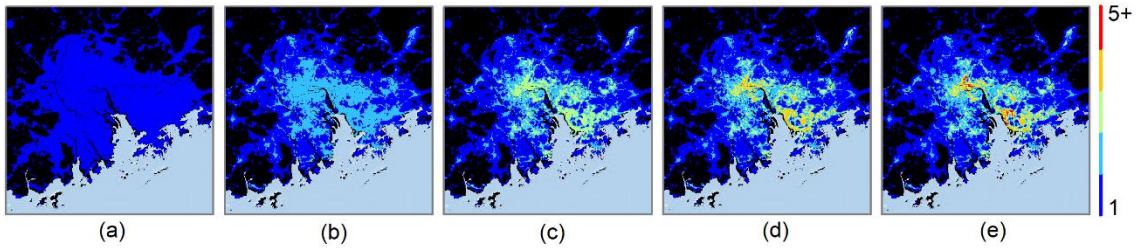

Figure 2: Zoom-in illustration of recursively generating natural cities around the world from the first iteration to the last iteration

(Note: Natural cities are recursively generated from the first iteration (a, in pure blue) to the last iteration (e, in purr red). There are six iterations in total to generate all natural cities from NTL imagery in 2022. However, natural cities generated at the last iterations are too few to be perceived. So natural cities generated later than the fifth iteration (5+ in the legend) are combined for the sake of better visualization. The continents and oceans are visualized in gray and navy blue, respectively.)

4) You are free to zoom in for detailed boundaries of natural cities and zoom out for glancing at all natural cities around the world at global scale (Figure 3). It is important to assign both the local and global views to an appropriate projected coordinate system to enhance the visual quality. Specifically, for the detailed view of the Greater Bay Area in China shown in Figure 2, the use of *WGS_1984_UTM_Zone_49N* is advised. For the global view presented in Figure 3, the *Eckert III*

projection with the central meridian at 150°E is recommended (Panel f to Panel g of Figure 1). For local view, appropriate UTM projected coordinate system should be selected.

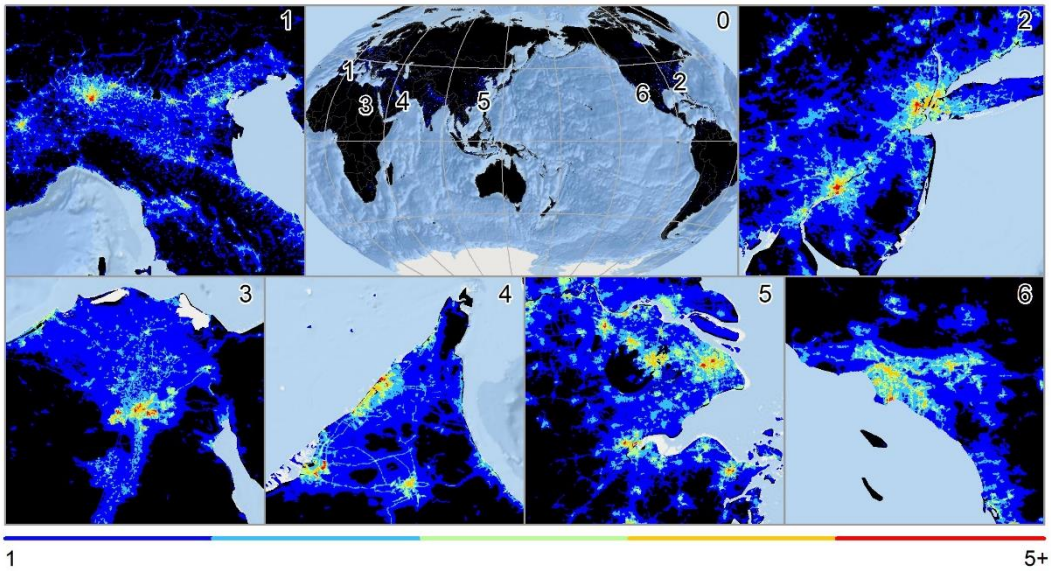

Figure 3: Global view of natural cities around the world and six typical regions of natural cities (Note: The six typical regions (Panel 1-6) are selected based on all natural cities generated around the world (Panel 0). The global view is using the Winkel Tripel projected coordinate system with its centra line at 150°E (Panel f to Panel g of Figure 1). Each panel, from 1 to 6, corresponds to distinct areas with significant: north Italy, the New York-Boston corridor, the Nile region, the United Arab Emirates, the Yangtze River Delta, and Los Angeles, which exhibit unique and complex morphologies of natural cities. It should be noticed that each location is recommended to select appropriate projected coordinate system, please refer to https://mangomap.com/robertyoung/maps/69585/what-utm-zone-am-i-in-# for choosing correct coordinate system.)

You are encouraged to explore more beautiful patterns from the natural cities worldwide and compare different natural cities from different areas. The qualitative analysis provides you with strong evidence and insights on how urban spaces are evolving and forming recursively. You are also encouraged to go through the source code to better understand how the recursive approach is implemented in detail and try to modify different parameters for more discussion on the sensibility of the algorithm.

4. Analyzing the results of complexity of natural cities

The degree of complexity is determined by the numbers of natural cities and their underlying scaling hierarchy across each iteration. It indicates that the more natural cities and higher levels of hierarchy, the more complex the urban spaces are. In other words, the world is organized more complicated in that case. The results of complexity will be output automatically during the process of generating natural cities through the Python 3 program. However, we suggest analyzing by your own self based on the result files under *Results_complexity* folder to get a more comprehensive understanding of the recursive definition of complexity. Taking the same NTL imagery selected in Section 3, you can follow the below steps the quantify the urban complexity:

1) Open the result csv file in Excel (Table 1), which each column represents the number of iterations(I), the amount of decomposable natural cities (D_NC), all natural cities (NC), the average degree of hierarchy of all natural cities (H) and the degree of complexity for this iteration (CR), respectively.

2) The generating process starts by decomposing the global NTL imagery. The results Table 1 show, the first iteration leads to 672,441 natural cities with average 12 levels of hierarchy, of which 2,505 are decomposable, leading to 104,909 natural cities with average 5.12 levels of hierarchy, 997 of which are decomposable, leading to 23,028 natural cities with average 3.83 levels of hierarchy, 360

of which are decomposable, leading to 6,971 natural cities with average 3.61 levels of hierarchy, 91 of which are decomposable, leading to 1,302 natural cities with average 3.26 levels of hierarchy, 15 of which are decomposable, leading to 123 natural cities with average 3 levels of hierarchy, none of which are decomposable.

3) The total degree of complexity is the sum up of the column of CR, which equals to 8,724,444 = 8,069,292 + 537,204 + 88,170 + 25,162 + 4,247 + 369. The total number of natural cities generated recursively equals to 6,860,745 = 672,441 + 104,909 + 23,028 + 6,971 + 1,302 + 123.

Table 1: Statistics of urban complexity of natural cities around the world
(Note: This table supplement Figure 3 to show the underlying statistics of natural cities. D_NC = decomposable natural cities, NC = natural cities; H = hierarchy of natural cities; CR = the degree of complexity. The underlined numbers of substructures are derived from multiple decomposable natural cities, so the corresponding H are calculated from CR/NC.)

Iteration	D_NC	NC	H	CR
1	1	672,441	12	8,069,292
2	2,505	104,909	5.12	537,204
3	997	23,028	3.83	88,170
4	360	6,971	3.61	25,162
5	91	1,302	3.26	4,247
6	15	123	3	369

Section 3 offers a geometric perspective, while Section 4 provides a statistical viewpoint on urban complexity within natural cities worldwide. In each iteration, the scaling hierarchy comprises far more small natural cities than large ones, contributing to the non-linearity and long-tailed distribution observed in urban complexity. Moreover, natural cities are recursively defined that can be decomposed iteratively until none remain decomposable. The recursive definition of cities indicates that urbanization is not a homogeneous, but rather a highly heterogeneous process, reflecting diverse factors and dynamics within cities.

5. Interpreting the temporal dynamics of natural cities

The living environment is undergoing continuous and rapid development through urbanization, with NTL data offering a comprehensive method to monitor the dynamics of human activities and infrastructure developments on the Earth's surface. The NTL products discussed in this tutorial span the last 23 years, starting from the year 2000 to 2022. By following Section 5, you can visualize the growing trend of urban complexity generated from NTL images of each year. It is to be noted that the entire procedure will take you about 30 minutes.

1) Reconduct generating the natural cities from NTL imagery for the entire 23 years following the steps in Section 3. Please make sure again your NTL images are named as *2000.tif*, *2001.tif*, …*2022.tif* (Panel b of Figure 1). You can comment out the line 51,52, 57 and 58 in *generate_NC.py*, to only output the statistics of complexity and reduce time consumption. The output statistics tables will be saved as *2000.tif*, *2001.tif*, …*2022.tif* (Panel b of Figure 1).

2) Run *interpret_dynamics.py* if you successfully obtain 23 sheets of statistics, which should have been saved as *2000.csv*, *2001.csv*, …*2022.csv* under *Results_complexity (*Panel d of Figure 1). The temporal dynamics will be generated automatically and saved as *dynamics.jpg* under the root folder (Figure 4).

3) If you're interested in the morphology dynamics of natural cities, you can keep these lines and try to visualize the changes following Section 3. As an alternative, you can try to modify the Python script to automatically repeat the identification of natural cities on NTL images during the last 23 years.

4) Meanwhile, you are also encouraged to visualize the dynamics of the number of natural cities or decomposable natural cities during the last 23 years, by either creating a double-y axis plot or creating a new line plot.

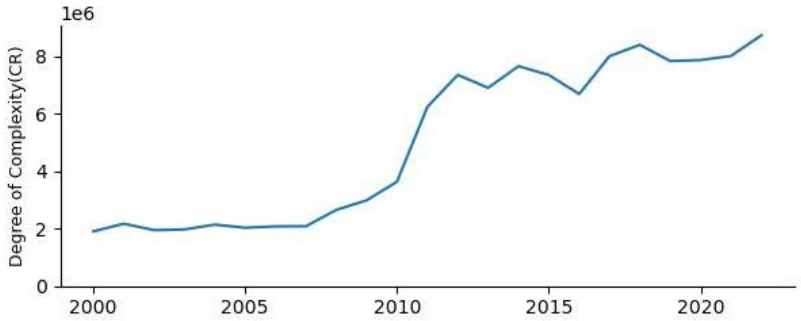

Figure 4: The evolving complexity of natural cities around the world
(Note: The line graph contains the last 23 years in total from 2000 to 2022.)

The line graph provides statistical evidence of the dynamics for urban complexity calculated from NTL imagery. The results show the Earth's surface is becoming more and more complex organized by numerous natural cities and indicate that there are increasing numbers of natural cities and their scaling hierarchies. During the initial seven-year period, there was a gradual increase in urban complexity. Then between 2008 and 2012, there was a notable acceleration in this trend. Subsequently, starting from 2013, urban complexity experienced a period of consistent and stable growth.

6. Conclusion

Natural cities provide organic insights for defining a city in a top-down manner from geographic big data, which is specifically the NTL imagery in this tutorial. This definition from big data aims to pursue a human-centered approach to urban governance, with the goal of constructing cities that can adapt to increasingly complex challenges while enhancing social cohesion. The key concept of urban complexity is lying under the scaling hierarchy of substructures, mirroring the recursive organization of the Earth's surface, as well as a living structure. The scaling hierarchy indicates that there are far more small natural cities than large ones. Based on it, the complexity is quantifiable by the number of natural cities and degree of hierarchical levels. In other words, the greater the number of natural cities and the higher levels of their hierarchy, the more complex the Earth's surface becomes. The recursive generation of natural cities is highly time-consuming. By applying connected components analysis, the recursive approach can be conducted in global NTL images within an acceptable time and supports the temporal analysis of the evolving complexity of urban areas on Earth's surface.

Acknowledgement

I would like to thank Prof. Bin Jiang for his patient guidance on this tutorial. The recursive approach of generating natural cities is developed the recursive approach to computing structural beauty of images and livingness of space (Jiang and de Rijke 2023). The Python 3 implementation is refactored based on the ArcPy implementation by Mr. de Rijke.

References:

Alexander C. (2002–2005), *The Nature of Order: An essay on the art of building and the nature of the universe*, Center for Environmental Structure: Berkeley, CA.

Alexander C. (2003), *New Concepts in Complexity Theory: Arising from studies in the field of architecture, an overview of the four books of The Nature of Order with emphasis on the scientific problems which are raised*, http://natureoforder.com/library/scientific-introduction.pdf.

Batty M. and Longley P. (1994), *Fractal Cities: A geometry of form and function*, Academic Press: London.

Chen Z., Yu B., Yang C., Zhou Y., Yao S., Qian X., Wang C., Wu B. and Wu J. (2021), An extended time series (2000–2018) of global NPP-VIIRS-Like nighttime light data from a cross-Sensor calibration,

Earth System Science Data, 13(3), 889–906.
Christaller W. (1933, 1966), *Central Places in Southern Germany*, Prentice Hall: Englewood Cliffs, N. J.
Jiang B., Yin J. and Liu Q. (2015), Zipf's law for all the natural cities a round the world, *International Journal of Geographical Information Science*, 29(3), 498–522.
Jiang B. (2013), Head/tail breaks: A new classification scheme for data with a heavy-tailed distribution, *The Professional Geographer*, 65(3), 482–494.
Jiang B. and de Rijke C. (2023), Living images: A recursive approach to computing the structural beauty of images or the livingness of space, *Annals of the American Association of Geographers*, 1–19.
Mandelbrot B. B. (1982), *The Fractal Geometry of Nature*, W. H. Freeman and Co.: New York.
Samet H. and Tamminen M. (1988), Efficient component labeling of images of arbitrary dimension represented by linear bintrees, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4), 579–586.

Appendix A: A natural city as a connected component of pixels

A natural city is identified as the brightest hotspot region in NTL imagery, or technically, it is a connected component of a cluster of pixels. Using a recursive approach involves repeatedly delineating inner hotspots within these larger natural cities. Therefore, it is highly time-consuming to the conduct recursive approach due to its repetition required in identification of the inner hotspots. On the other hand, there are large amounts of natural cities on a global scale which demands a high vacancy of computer memory spaces. The conventional approach to identifying natural cities from NTL imagery is based on the vectorization of binary images with value 1 representing bright or urban pixels and value 0 representing dark or rural pixels (Jiang et al. 2015), which tries to convert the raster to vector composed of numerous points and edges. Although vector will provide more details about boundary than raster, the vectorization will waste large amounts of memory spaces and increase the computing time. In order to enhance the efficiency of identifying natural cities on a global scale and reduce the occupation of computer memory space, the recursive approach regards a natural city as a connected component of pixels instead of a vector polygon (Figure A1).

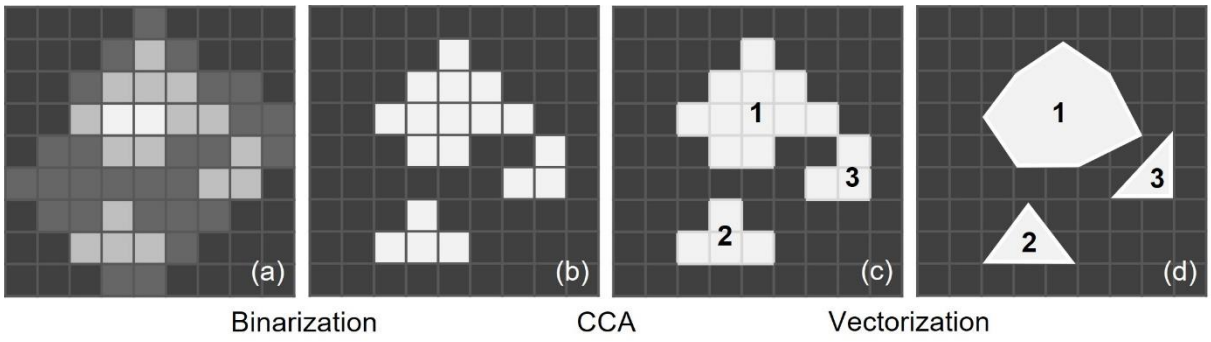

Figure A1: Illustration of raster-based and vector-based approach to delineating natural cities (Note: CCA denotes the connected component analysis. In order to identify natural cities, the NTL imagery (Panel a) is binarized by the average value of brightness, where the value of 1 represents urban areas, and a value of 0 represents rural areas (Panel b). By CCA, three different regions of bright pixels are identified maintaining the raster format (Panel c). The traditional approach will further convert the connected regions into vectors (Panel d), thereby the process is much slower than the new implementation.

Connected Component Analysis (CCA) is to cluster a set of connected pixels from binary images (Samet and Tamminen 1988). It will label all the pixels in a detected component as the same value to identify different clusters or components. The process of generating connected components maintains the raster format of natural cities, aligning with the structure of NTL imagery. This approach is notably more computationally efficient compared to vectorizing natural cities. There is no requirement to store a

separate vector file for each natural city, resulting in significant savings in computer storage space. For example, when CAA, the detection of 3 natural cities (Panel b) results in their representation as 3 distinct pixel-based connected regions (Panel c), rather than converting them into continuous areas represented by polygons (Panel d). By assigning 3 distinct values, the connected component analysis can successfully delineate different natural cities from NTL imagery and avoid converting them into different formats like vector polygon. Thereby, it ensures significantly higher computational efficiency and reduces space occupation compared to the conventional approach, resulting in at least a tenfold reduction in processing time.