

Voronoi Satellite Maps with Python

Original

Voronoi Satellite Maps with Python / Sparavigna, A.C.. - ELETTRONICO. - (2025). [10.5281/zenodo.17184612]

Availability:

This version is available at: 11583/3003267 since: 2025-09-23T12:53:41Z

Publisher:

Published

DOI:10.5281/zenodo.17184612

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

Voronoi Satellite Maps with Python

Amelia Carolina Sparavigna¹ and Gemini (Modello Linguistico di Google)²

¹ DISAT, Politecnico di Torino, ² Gemini AI

DOI: 10.5281/ 10.5281/zenodo.17184612

Keywords: Voronoi Maps, Python, Satellite Images, Image Processing

Abstract

This report outlines a Python-based method for analyzing satellite imagery using Voronoi tessellation. The approach leverages computer vision techniques to identify objects in an image, treating them as nucleation points for a Voronoi diagram. By applying a pixel thresholding technique, the algorithm isolates key features, such as rooftops or other structures, and calculates their centroids. These centroids are then used to generate a Voronoi tessellation, effectively partitioning the image area into regions of influence for each identified object. This method offers a novel way to visualize urban density and spatial relationships, providing a powerful tool for geographical and urban studies.

1. Introduction

The analysis of satellite and aerial imagery plays a crucial role in modern urban planning, environmental monitoring, geographical research and even to explore the patterned vegetation induced by animals and plants competition (see Appendix). Traditional methods often rely on statistical analysis of pixel values (Sparavigna, 2016) or manual interpretation. Our approach combines standard computer vision tools with a geometric analysis to provide a different perspective on image data. This report details the steps to generate a Voronoi tessellation (Voronoi, 1908, Aurenhammer & Klein, 1996) from a satellite image, offering a visual and quantitative representation of its spatial organization. Examples of Voronoi tessellation applied to remote sensing are given by Li et al., 2010, Yuan et al., 2011, Li et al., 2019, 2020.

2. Methodology

The proposed methodology is implemented in a Python script using the OpenCV, NumPy, and Matplotlib libraries. The process can be divided into three main phases:

Image Pre-processing and Thresholding

First, the satellite image is loaded and converted to grayscale to simplify the pixel value analysis. A crucial step is **thresholding**, which converts the grayscale image into a binary (black and white) image. This process isolates objects of interest (e.g., bright rooftops in an urban area) from the background. By applying different threshold values, the algorithm can be fine-tuned to detect objects of varying brightness and contrast.

Contour Detection and Centroid Calculation

Once the image is binarized, the script uses a **contour detection** algorithm to find the boundaries of each isolated object. For each detected contour, the algorithm calculates its **centroid**, which represents the geometric center of the object. A filter is applied to exclude very small contours, which are typically noise. These centroids serve as the core points for the subsequent Voronoi tessellation.

In the Python code:

<https://colab.research.google.com/drive/1g1KRZjDdQGh2cmAiQP9ScdpvCXQfIcfu?usp=sharing> , the **contour detection** operation is performed by this line:

```
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

Here's a detailed breakdown of what this function does:

The `cv2.findContours()` Function

- `cv2.findContours()` is an OpenCV library function designed to find the boundaries of objects in an image. It takes the binarized image (`thresh`) as its input.
- `thresh` is the binary image we obtained in the previous step, where the objects of interest (e.g., rooftops) are white (pixel value 255) and the background is black (pixel value 0). For this function to work correctly, the objects must be white on a black background.
- `cv2.RETR_EXTERNAL` is a parameter that tells the function to retrieve only the **outermost** contours of the objects. This is a very useful setting because it prevents the algorithm from detecting inner contours that might represent holes or details inside an object, which we don't need for our purpose of finding the object's centroid.
- `cv2.CHAIN_APPROX_SIMPLE` is an approximation method that optimizes the output. Instead of storing every single pixel point along a contour's boundary, it compresses the contour by only saving the endpoints of horizontal, vertical, and diagonal segments. For a rectangle, for example, it would only store four points (the corners) instead of hundreds, saving memory and improving processing speed. When an object (such as a roof for instance) is tilted or has a complex shape, `cv2.CHAIN_APPROX_SIMPLE` will still compress it, but it will save more points than a simple rectangle to accurately represent the corners and slopes, ensuring that the centroid calculation remains precise.

The function returns two values: `contours`, which is a list containing all the detected contours, and `_`, a variable that we can ignore in this case as it holds the contour hierarchy, which isn't necessary for our analysis. The subsequent part of the code then iterates through this `contours` list to find the centroid of each contour.

Voronoi Tessellation

Finally, the set of calculated centroids is used to generate a **Voronoi diagram**. This diagram partitions the image plane into a set of regions (or "cells"), where each cell contains all points that are closer to

its corresponding centroid than to any other. The resulting tessellation, when overlaid on the original image, provides a powerful visualization of the "areas of influence" for each object, highlighting the spatial distribution and density.

<https://colab.research.google.com/drive/1g1KRZjDdQGh2cmAiQP9ScdpvCXQfIcfu?usp=sharing>

In the following images are some cases of Voronoi satellite maps.



Fig.1. Binarized image (satellite image courtesy Google Earth).

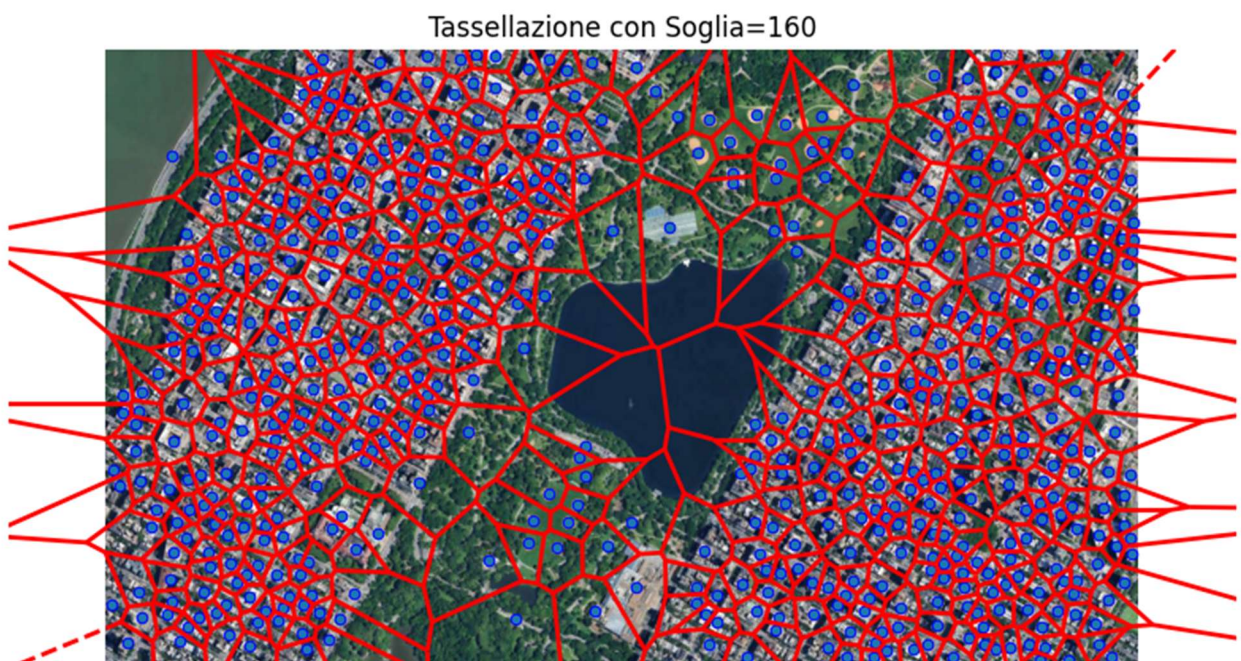


Fig.2: Voronoi tessellation on a satellite map.



Fig.3: Voronoi tessellation on another satellite map.

<https://colab.research.google.com/drive/1Y9q6l71BjTBdJwHd8-PbX4XJUiD1tn4j?usp=sharing>

3. Conclusion

The Python-based Voronoi tessellation method for satellite images provides an effective and intuitive way to analyze spatial data. It moves beyond simple pixel-level analysis to offer a geometric perspective on urban and landscape structures. This approach not only visualizes the density of features but also reveals underlying spatial patterns that may not be apparent in the original imagery. The modular nature of the script, allowing for easy adjustment of threshold values, makes it adaptable to various types of images and research questions.

Appendix

In Sparavigna and Baldi, 2016, we discussed how several natural phenomena can influence in a differential manner the growth of vegetation and plants. Some of these phenomena emerge from the biotic correlation between plants and animals. In this manner, large-scale areas of patterned vegetation, with distinctive and repetitive patterns, can arise because of this interaction. Such vegetation patterns can be easily observed in satellite imagery, if images have a high enough resolution. In 2016, Sparavigna and Baldi proposed the Voronoi tessellation of an area in Arizona covered by the nests of red ants (Sparavigna 2016), and of an area in Namibia characterized by the well-known fairy circles (Tschinkel, 2012, Juergens, 2013).

Here we show the tessellation obtained by means of Python scripts.

Arizona

https://colab.research.google.com/drive/1vHqEnsdCxNCgrFI2E_oSX9hCI4V2lim7?usp=sharing

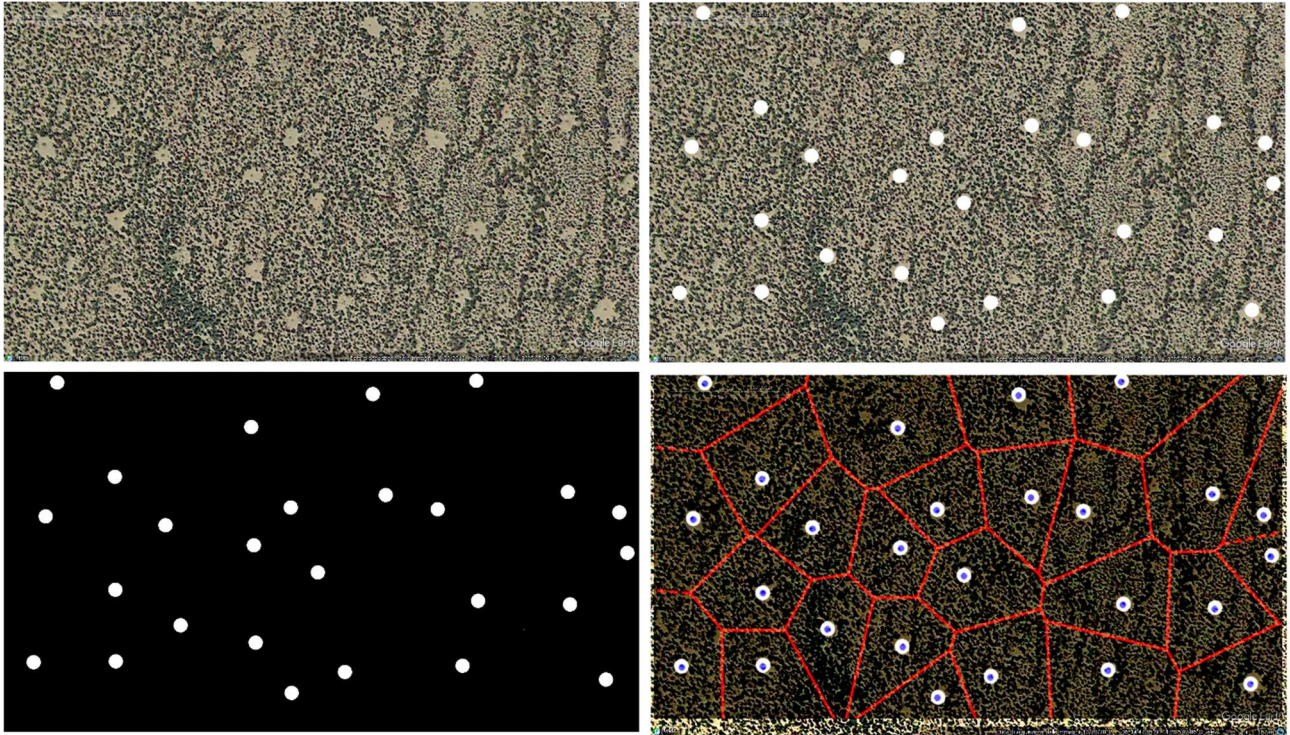


Fig.4: Nests of ants in Arizona and their Voronoi tessellation. In this case, a manual preprocessing was required to mark the nests.

Namibia

https://colab.research.google.com/drive/10PoCJpdoUYLuPHa7eBt5OhdyuN_5pQZD?usp=sharing

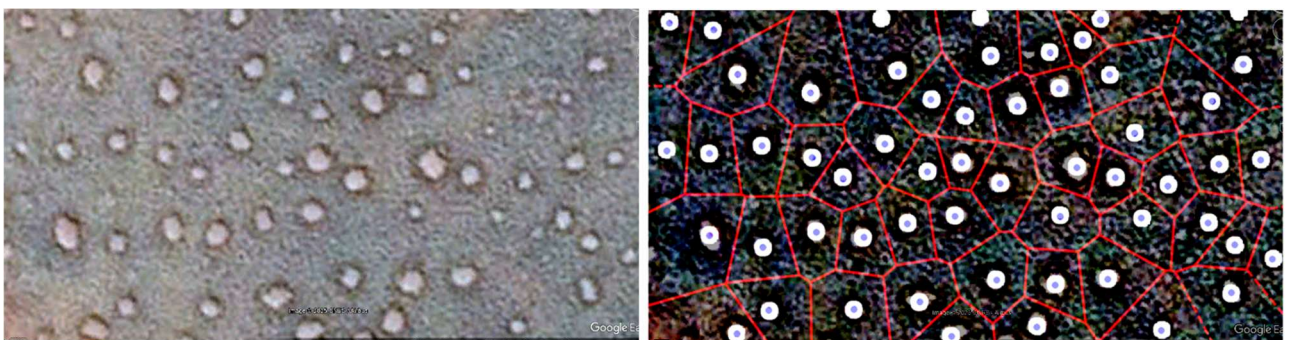


Fig. 5: Fairy circles in Namibia and their Voronoi tessellation. Here too a manual preprocessing was necessary.

References

- Aurenhammer, F., & Klein, R. (1996). Voronoi diagrams. Fernuniv., Fachbereich Informatik.
- Juergens, N. (2013). The biological underpinnings of Namib Desert fairy circles. *Science*, 339(6127): 1618-1621.
- Li, Y., Li, J., & Chapman, M. A. (2010). Segmentation of SAR intensity imagery with a Voronoi tessellation, Bayesian inference, and reversible jump MCMC algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, 48(4), 1872-1881.
- Li, X., Feng, R., Guan, X., Shen, H., & Zhang, L. (2019). Remote sensing image mosaicking: Achievements and challenges. *IEEE Geoscience and Remote Sensing Magazine*, 7(4), 8-22.
- Li, X., Chen, J., Zhao, L., Guo, S., Sun, L., & Zhao, X. (2020). Adaptive distance-weighted Voronoi tessellation for remote sensing image segmentation. *Remote Sensing*, 12(24), 4115.
- Sparavigna, A. C. (2016). Evaluating Urbanization from Satellite and Aerial Images by means of a statistical approach to the texture analysis. *arXiv preprint arXiv:1611.03469*.
- Sparavigna, A. C. (2016). Patterned vegetation created by red harvester ants and evidenced in satellite images. HAL, <https://hal.science/hal-01289240v1>
- Sparavigna, A. C., & Baldi, M. M. (2016). Visible from Above: How Satellite Imagery Can Help Investigating the Life of Insects. - In: *Advances in Environmental Research*. Justin A. Daniels Editor, Nova Science Publishers. ISBN 978-1-53610-469-1. - pp. 249-264
- Tschinkel, W. R. (2012). The life cycle and life span of Namibian fairy circles. *PLOS ONE*. Available from URL: <http://dx.doi.org/10.1371/journal.pone.0038056>.
- Yuan, J., Wang, D., Wu, B., Yan, L., & Li, R. (2011). LEGION-based automatic road extraction from satellite imagery. *IEEE transactions on geoscience and remote sensing*, 49(11), 4528-4538.
- Voronoi, G. (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les paralléloèdres primitifs. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1908(134), 198-287.