International Journal of Data Science and Analysis
2022; 8(4): 119-130
http://www.sciencepublishinggroup.com/j/ijdsa
doi: 10.11648/j.ijdsa.20220804.12
ISSN: 2575-1883 (Print); ISSN: 2575-1891 (Online)

Mapping and Quantifying Green Area of Urban Villages from Google Satellite Images Using *k*-Means Clustering Algorithm

Veronica Sri Moertini*, Fritz Humphrey Silalahi

Department of Informatics, Parahyangan Catholic University, Bandung, Indonesia

Email address:

moertini@unpar.ac.id (Veronica Sri Moertini), fritz.humphrey11@gmail.com (Fritz Humphrey Silalahi)

*Corresponding author

To cite this article:

Veronica Sri Moertini, Fritz Humphrey Silalahi. Mapping and Quantifying Green Area of Urban Villages from Google Satellite Images Using *k*-Means Clustering Algorithm. *International Journal of Data Science and Analysis*. Vol. 8, No. 4, 2022, pp. 119-130.
doi: 10.11648/j.ijdsa.20220804.12

Received: July 29, 2022; **Accepted**: August 30, 2022; **Published**: September 14, 2022

Abstract: It is well known that trees in cities are important as they produce oxygen, absorb rainwater, and provide shades. Due to urban development, the trees sufficiency in many cities is threatened. To maintain cities healthy environment, it is imperative that the local authorities monitor the trees adequacy from time to time and then act accordingly. For doing so, they need to be supported with quantitative data representing the adequacy of trees that can easily be accessed. In Indonesia, the lowest level of governmental administrative area is urban village. We view that if the data is provided for this level, it would be more effective as the head of the urban village can act or create necessary programs accordingly. Google updates regularly and provides satellite images that can be freely downloaded with many zoom-level. In urban areas, trees are visible with zoom-level of 16 and beyond. This research aims to develop a method for detecting/mapping trees green areas from urban village satellite images with the final result of green area (approximated in hectare unit). The method include village images preparation, detecting green areas based on image segmentation approach (using k-Means clustering algorithm), mapping and computing green area for each village. The case study is Bandung city, which is one of the most populated cities in Indonesia. The findings are potentially be used by the local authorities to evaluate the adequacy of trees in their territories.

Keywords: Satellite Image Segmentation, Green Area Detection and Mapping, Urban Village Green Area

1. Introduction

It is well known that trees produce oxygen and absorb CO_2 for photosynthesis during the day, thereby contributing to reducing the harmful effects of CO_2. Trees also provide shade and absorb rainwater. However, due to industrialization and residential area development in densely populated urban areas, trees have been decreasing significantly. This may cause negative impacts on various aspects, including the occurrence of floods and increasing air pollution. Therefore, the adequacy of trees should be maintained from time to time by the local authorities.

By looking at the satellite imagery on Google Map (https://www.google.com/maps), our eyes can identify which areas are overgrown with trees, rice fields, buildings, roads, and other objects. Due to the fact that human eyes recognize objects in the image based on their color composition, we may consider that the objects have the potential to be "recognized" based on their color by utilizing image analysis approaches, such as clustering or classification algorithm. Fortunately, Google provides satellite images which can be free downloaded using a built-in API. The user can select a satellite image at a certain zoom level as needed and download the image in the form of tiles or image partitions (see Section 2.2).

Satellites aimed to capture data from earth use many bands of wavelength and produce many bands of data, such as Red, Green, Blue, NIR (Near Infra Red), Panchromatic, Thermal Infrared, etc. There are several sources that publish data from satellites. USGS provides satellite images of LANDSAT 5, 7, 8 and Sentinel 2A [15]. European Union publishes Sentinel-3 satellite data [17]. Several websites also publish satellite data,

such as Gaofen-2 satellite data [18]. Remote sensing researchers analyze combination of those bands for many purposes, such as land usage distribution [15], drought mapping [17], and flood monitoring [13]. The techniques used can be complex, which are mastered by remote sensing specialists. Hence, this research aims to analyze satellite data with digital image processing approach, specifically image segmentation technique, using a clustering algorithm that is widely used by data scientists.

Image segmentation is a technique that is often used to partition an image into several areas based on the characteristics of the pixels in the image [7]. It separates the foreground from the background or clustering pixel areas based on similarity in color or shape. Image segmentation can be classified into edge detection-based, threshold values based, area-based, feature-based and artificial neural networks. Semantic segmentation of satellite and aerial imagery has several uses including automated map generation, land use analysis, urban planning, etc. [5].

Machine learning algorithms have been utilized for segmenting images, which can be based on unsupervised learning, namely clustering algorithms. Examples of cases can be found in [1, 3, 9, 12]. The segmentation aims to group areas that have a certain pattern into the same group. When the segmentation is performed based on the image color, the results are clusters of area where each cluster has similar colors and is relatively different from other clusters. In the context of satellite images, these clusters can mean residential areas, roads, trees, vacant land, etc.

Detection of land use in satellite images (e.g. in urban areas) can also be conducted using classification techniques, such as deep learning which is currently widely used. To train deep learning (e.g. Convolutional Neural Network), ground-truth data that has been labeled with the targeted classes is needed [2, 19]. Example of classes are settlements, gardens/forests, rice fields, etc. The preparation of ground-truth data, which must be in large quantities for the model to be trained properly, requires considerable effort and cost. On the other hand, image analysis using the clustering algorithm does not require ground-truth data, but only satellite images. Although in general the classification model can be of better quality (e.g. more accurate or precise), but for certain purposes that do not require high accuracy, clustering techniques can be utilized.

The results of image segmentation can be used to map and quantify trees in urban areas. To support effective decision making from the lowest to the upmost level authorities in cities, the area to be mapped and quantified should be based on the governmental administrative area. In Indonesia, the smallest unit of administrative area is urban village (*kelurahan*). Currently, villages boundaries (in GeoJSON formats) in lots of cities are available for download. The boundaries can be processed to produce the coordinates of tiles on a Google map, then the tiles of the village can be downloaded, processed and analyzed using segmentation approach, to detect the green area of trees and calculate its area.

The objective of this research is to develop a method for detecting/mapping trees green areas from urban village satellite images with the final result of approximated green area (in hectare unit). The trees being detected are those that grow in residential areas, roads, and gardens/forests (see Figure 1). The detection uses a clustering algorithm, which is *k*-Means, so it does not require ground truth data. Based on our literature study, no similar methods have been developed, especially detecting trees from the satellite image of the smallest governmental administrative area. Thus, it is expected that this research has a scientific contribution. The case study is Bandung city, which is one of the most populated cities in Indonesia, located in West Java (Figure 2).

Figure 1. *The green areas.*

The results of this study are expected to be used by the government (regional and/or central) for the purposes of monitoring trees area in cities from time to time to support decision making or designing policies for maintaining the adequacy of trees in cities.

2. Related Works

The availability of free and historical satellite images has led to many research activities, specifically for analyzing

those images with the purpose of land cover classification or segmentation as well as monitoring over the years. Below are discussions of four research results related to our proposed methods.

Satellite images are always partitioned into small regular patches. Semantic segmentation using deep neural networks (DNNs) commonly individually fed those patches into the DNN. The underlying assumption is that these images are independent of one another in terms of geographic spatial information. However, it is well known that many land-cover or land-use categories share common regional characteristics within a certain spatial scale. To address the issue, Yang, N. & Tang, H. [19] explored deep learning approaches integrated with geospatial hash codes (GHC) to improve the semantic segmentation results of regional satellite images. The geographic coordinates of satellite images are encoded into a string of binary codes using the geohash method. The binary codes of the geographic coordinates are fed into the deep neural network using three different methods in order to enhance the semantic segmentation results. Experiments were conducted using GID dataset, which consists of 150 images collected from the Gaofen-2 satellite. Each image has a pixel size of 6908 x 7300 and contains the R, G, and B bands. The GID dataset contains five land-use classes: built-up, farmland, forest, meadow and waters. The experiments demonstrate the effectiveness of the proposed approach.

Figure 2. *The location of the case study city, Bandung, in Indonesia.*

Semantic segmentation of satellite and aerial imageries has many applications including automated map making, land use analysis, urban planning and so on. Favorskayaa, M. N., Zotina, A. G. [5] develops a special type of semantic segmentation. The method is as follows: (1) The boundaries of natural objects such as agricultural fields are detected and approximated by polygons. (2) Texture recognition based on Digital Wavelet Transform (DWT) and Local Binary Patterns (LBPs) using a limited textural dictionary are performed on the polygons. (3) The extracted features are embedded into the images as watermarks, which are then used to give labels to each of the polygons (such as wheat, corn, clear, etc.).

In planning and reviewing changes in the ground overview data, land changes distribution identification are critical. The free satellite images offers a valuable resource to be continuously and accurately mapped and tracked year by year. Tin, S. N. and Muttitanon, W. [16] develop methods to extract the built-up area for the urban planning area every five years from the satellite images of LANDSAT 5, 7, 8 and Sentinel 2A from USGS. GIS tools such as raster calculator and built-up region modeling are used to measure the indices that include the Enhanced Built-up and Bareness Index (EBBI), the Normalized Difference Built-up Index (NDBI) and the Urban Index (UI) or the Built-up Index (BUI). The study area is Yangon, a commercial capital of Myanmar, which has been developing rapidly. The proposed methods are used to analyze the LANDSAT images from 1990 to 2020. The results indicate that those can be used to track the Yangon changes year by year.

One of the most common hydro-meteorological disasters is the drought that occurs every year in the dry season. This disaster causes crop failure, land and forest fires, and clean water shortages. In [18], the Sea and Land Surface Temperature Radiometer (SLSTR) instrument onboard the Sentinel-3 platform was used to map drought using the Vegetation Temperature Condition Index (VTCI) algorithm, which is based on the scattering space technique of Land Surface Temperature (LST) and Normalized Different Vegetation Index (NDVI). The case study is Java Island in Indonesia. Twenty-two SLSTR Sentinel-3 data, recorded every two times a month from January to December 2018, was analyzed for drought monitoring based on the scattering of LST and NDVI values. The method is as follows: (1) Pre-processing the dataset, including data sub-setting, re-projection, and filtering to produce NDVI and LST. (2) Scatter-space of NDVI and LST were produced, in order to observe the correlation between them. (3) Two linier regressions are applied to the NDVI and LST values, which are at top edge (warm) and bottom edge (cold). The coefficient of the two liner regressions were used to produce LSTNDVI_max and LSTNDVI_min, which then used to map the drought using VTCI algorithm. The research concludes that drought monitoring can be performed using the proposed methods.

As presented on the above discussions, none have analyzed the satellite data/images on the lowest level of governmental administrative area (urban village) and developed methods accordingly. Our proposed methods, therefore, aim to analyze images of urban village such that the results can be used to

monitor, evaluate and support the environmental program or decision making for the lowest level governmental authorities and beyond.

3. Methodology

3.1. Google Satellite Images Dataset

Google provides satellite images of the earth at various zoom levels. The larger the zoom level, the smaller the ratio of the image area to the actual area on the earth. The general level range is 0 to 19. At level 0, the image displays a map of the whole world, while at level 19 the image shows objects on the earth quite clearly (e.g. buildings, trees, etc.).

Google provides chunks of satellite images, called tiles, each tile is of 256 x 256 pixels and can be downloaded by utilizing its API. For satellite images in a certain area (on earth), the greater the level used, the more tiles in the image. For instance, at zoom level 2, the satellite image is divided into 16 tiles, each tile can be identified by a coordinate pair (x,y), such as (0,0), (0,1),..., (3,3). The larger the zoom level, the more tiles in the satellite image. The tile dimensions at each zoom level are expressed by Formula 3 [10] (Map, 2022).

$$dimTiles = 2^{level\_zoom} \quad (1)$$

Based on Formula 1, if the zoom level = 2, then *dimTiles* = $2^2 = 4$ so that the tile coordinates are (0…3, 0…3) and a total of 4×4 tiles or 16 tiles. If zoom level = 3, then dimTiles = 8, and number of tiles = 64.

The world coordinates on Google Maps are measured from the origin of the Mercator projection (northwest or top left corner of the map with a longitude of 180 degrees and an approximate latitude of 85 degrees), then incremented in the x direction (to the right or east) and the y direction (downward or south), as shown in Figure 3.

On Google Maps, in addition to latitude and longitude, location points are given with positive and negative decimal degrees. Location points are also given with decimal degrees of latitude and longitude which are defined by Google and are always positive.

Figure 3. *The Google Maps coordinates are increased in x (horizontal) and y (vertical) direction*[1].

1 https://developers.google.com/maps/documentation/javascript/coordinates [accessed 23 July 2022].

The world coordinates on each tile have a range (0-255) for x and (0-255) for *y*. In addition to the world coordinates, Google adds tile coordinates and pixel coordinate information at a specific location point in the downloaded satellite image, with Formula 2.

$$pixelCoordinate = worldCoordinate \times 2^{zoomLevel} \quad (2)$$

Based on Formula 2 if the zoom level is 0, then the pixel coordinates on the map image are the same as the world coordinates. The formula also states that each increase in the zoom level results in a doubling of the pixel coordinate values (2^1).

Thus, every 1 level increase, produces an image with a resolution of 4 times the resolution of the image at the lower level. For example, at zoom level 1, a map image contains 4 tiles. As each tile is 256×256 pixels, the entire image map is (256+256)×(256+256) or 512×512 pixels, then each pixel can be referenced with values between (0,0) to (511,511). At zoom level 4, each pixel can be referenced with values between (0,0) to (256×24-1, 256×24-1), at zoom level 5 each pixel can be referenced with values between (0,0) to (256×25-1, 256×25-1).

In general, at zoom level *n*, each pixel can be referenced with values between (0,0) to (256×2^n-1, 256×2^n-1).

3.2. GeoJSON Format

GeoJSON is an open standard geospatial data interchange format that represents simple geographic features and their non-spatial attributes. Based on JavaScript Object Notation (JSON), GeoJSON is a format for encoding a variety of geographic data structures. It uses a geographic coordinate reference system, World Geodetic System 1984, and units of decimal degrees[2].

GeoJSON supports the following feature types:

1) Point (including addresses and locations);
2) Line string (including streets, highways, and boundaries);
3) Polygon (including countries, provinces, and tracts of land);
4) Multipart collections of point, line string, or polygon features.

Example of a simple GeoJSON file is as follows:

```
{ "type": "FeatureCollection",
    "features": [
      {"type": "Feature",
        "geometry": {
          "type": "Polygon",
          "coordinates": [
            [[100.0,0.0],[101.0,0.0],[101.0,1.0],
             [100.0,1.0],[100.0,0.0]]
            ]
        },
        "properties": {
```

2 https://doc.arcgis.com/en/arcgis-online/reference/geojson.htm [accessed 23 July 2022].

```
            "Id": "Id0001",
            "Location": {"Street": "A street name"}
            }
          }
        ]
    }
```


The boundaries of an urban village can be represented as a GeoJSON file with the coordinates field contains lots of pair of world coordinate. An example of GeoJSON for Cipaganti village in Coblong district and Bandung city is (the coordinates are not fully included):

{"type":"FeatureCollection","features":[{"type":"Feature", "properties":{"FID":4227,"OBJECTID":4228,"ID_DESA":" 3273230001","DESA":"CIPAGANTI","KECAMATAN":"C OBLONG","KABUPATEN":"K_BANDUNG","PROVINSI" :"JAWA BARAT",

"ID_KEC":"3273230","ID_KAB":"3273","ID_PROV":"3 2","Shape_Leng":5380.389716999999,"Shape_Area":68824 7.2141349999,"Luas_Km_":0.688247},"geometry":{"type":" Polygon","coordinates":[[[107.60627600026157,-6.8843799 99803662],[107.60654900014273,-6.884591999582912],[10 7.60668400017732,-

.....

[107.60624999980186,-6.884356999993794],[107.606276 00026157,-6.884379999803662]]]}}]}

The GeoJSON dataset can be visualized with the GeoJSON Map Viewer on an Internet browser, such as provided at the URL https://geojsonviewer.nsspot.net/.

Google provides satellite images as tiles and for downloading tiles using its API, indices of tiles need to be provided by the downloader. Hence, for downloading an entire tiles covering an urban village, the boundaries coordinates defined in a GeoJSON should be converted into tile indices, which are needed by the API.

3.3. k-Means Clustering Algorithm and Elbow Method

Cluster analysis towards dataset aims to group objects based on their similarity. Objects in one cluster are similar to each other, and less similar to objects in other clusters. One of the well-known algorithms used for image clustering or segmentations is k-Means [3, 9, 12].

The k-Means algorithm partitions a set of n vectors x_j, j = 1,…, n into c clusters G_i, i = 1,…, c, and finds the cluster center of each cluster, such that the cost function, J, is minimized [6]. If the distance function expressed as $d(x_k,c_i)$ is applied to x_k in cluster i, then the cost function J is:

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c}\left(\sum_{k,x_k \in G_i} d(x_k - c_i)\right) \quad (3)$$

The results of clustering are usually expressed by the membership matrix U(m), where m = the number of objects in the data set. U(i) contains the group number of the i-th object. Cluster center (centroid) c_i is the mean of each attribute value of all vectors that are grouped in cluster i:

$$c_i = \frac{1}{|G_i|}\sum_{k,x_k \in G_i} x_k \quad (4)$$

with $|G_i|$ = number of objects in G_i.

The steps of k-Means algorithms is described below.

Algorithm: k-Means

Input: m vector x_j, j = 1,…, n or $x_j = (a_1, a_2, \ldots a_n)$ and cluster number k.

Output: membership array, U(m), and centroids, $c = \{c_1, c_2, \ldots, c_k\}$, $c_i = (ca_1, ca_2, \ldots, ca_n)$

Steps:

1) Each cluster center c_i and cost function J is initialized with certain values.
2) Calculate matrix U, where $U(s)$ = the s^{th} object group number. The object s is grouped into p, if the distance of object s to the center of the cluster cp is the smallest (compared to the distance of object s to the centers of other clusters).
3) Calculate J using Formula 3. Clustering is stopped if the difference in the value of the current J iteration with the previous J iteration is less than a certain threshold value (e.g. 0.01), which means that the membership of objects in the clusters is (relatively) stable.
4) Compute the cluster center with Formula 4, then return to step (2).

The performance of the k-Means algorithm depends on the cluster center initialization values. The k-Means algorithm is quite efficient, with a complexity of $O(tkm)$, where m = number of vectors/objects, k = number of clusters, and t = number of iterations. Generally: $k, t << m$.

A selected clustering algorithm applied to a particular data set must produce high quality clusters, which are characterized by high similarity between objects in one cluster and low similarity to objects in different clusters.

As described above, the k-Means algorithm requires an input of the cluster number (k). To find the best number of clusters in a particular data set, it is necessary to conduct experiments aiming to find the best k. Clustering with the best k would generate clusters of good quality.

The Elbow method is one method that can be used to find the best k [15]. The idea of this method is to compare the inertia value of the clustering results for each k value. Inertia is the sum of the distances from the object to its cluster center (Formula 5).

$$Inertia = \sum_{k=1}^{C}\sum_{i=1}^{N}(x_i - c_k)^2 \quad (5)$$

where C = number of clusters, N = number of objects, x_i = object i, c_k = cluster center where object i belongs.

At a value of k that is smaller than the best (optimal) k, the Inertia is relatively large because adjacent objects are still possible not to be grouped in the same cluster. The value of Inertia decreased to the best k, after that the Inertia did not decrease as much as the previous decrease. Grouping objects with a value of k greater than the best value of k will result in the splitting of one or more clusters (which are already good) into smaller clusters, so that the inertia will not significantly decrease. On a graph of Inertia vs k, the best k will be at the elbow of the Inertia graph. In Figure 4, the best k (5) is marked

with a blue cross.

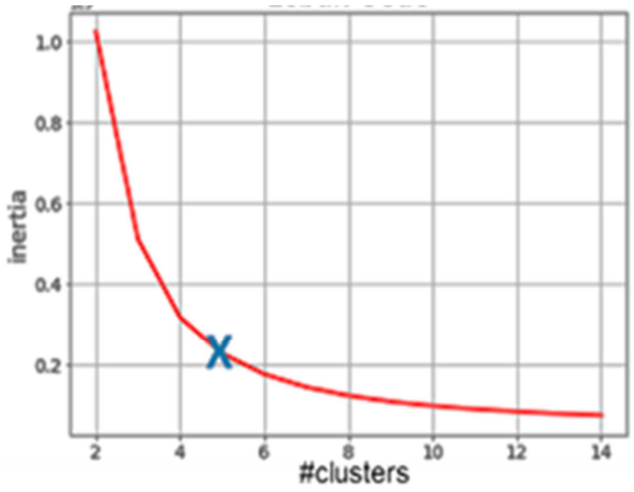

Figure 4. *Inertia graph with elbow at k = 5.*

3.4. Proposed Method

The proposed method consists of two major stage, which are:

Stage-1: Village image preparation (Figure 5). The urban village boundaries, which stored as GeoJSON file, is used to compute the Google tiles coordinates for zoom level of 16 (trees are visible with this level), which cover the urban village. The tiles coordinates are then used to download tiles accordingly, which are then merged to form a square image. Using the GeoJSON file, this image is then cut to form the village image.

Stage-2: Detecting, mapping, computing green area of the village and its percentage (Figure 6). The village image pixels are then clustered using *k*-Means algorithm using $k = 5$ (this k is obtained via experiments), the pixels' color that are grouped as trees are then converted into green. Then the area (in hectare) is computed along with its percentage.

The detailed discussion of each stage is presented below.

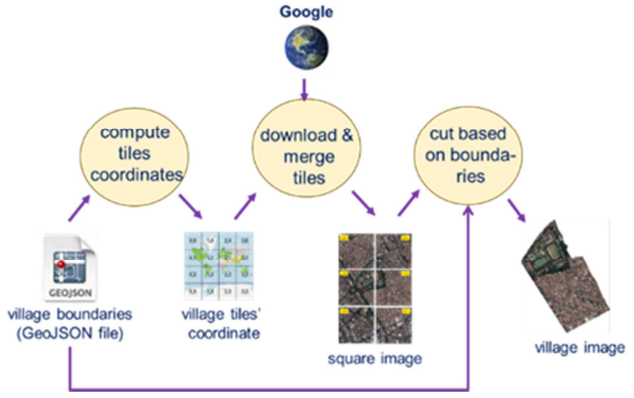

Figure 5. *Stage-1: Village image preparation.*

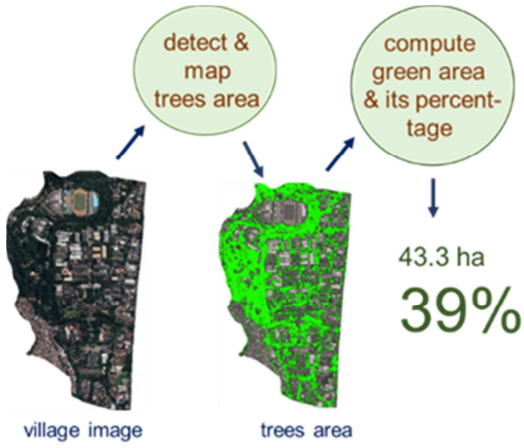

Figure 6. *Stage-2: Detecting, mapping and computing green area.*

Stage-1: Village image preparation

The steps of computing tiles coordinates, downloading and merging the tiles are discussed below:

(1) Read the dataset of village boundaries coordinates: The dataset of the boundary coordinates of a village in GeoJSON format is read and stored in the *data_koord []* array (where each array element is of type earth coordinates, pairs of latitude and longitude values).

(2) Find the min-max coordinates of the village boundary at longitude and latitude: From the *data_koord* array, look for the 4 outermost coordinates of the village or the minimum and maximum values for the longitude (*long*) and latitude (*lat*) coordinates, namely *T1* (*latmax, longmin*), *T2* (*latmax, longmax*), *T3* (*latmin, longmin*), *T4* (*latmin, longmax*) and then stored to an array, A. So, A = {*T1, T2. T3, T4*}.

For example, Figure 7 shows the outermost coordinate points of the Sukamaju village.

Figure 7. *The outer most coordinates of Sukamaju village.*

(3) Convert min-max coordinates to tile coordinates: By

using Formula 2, at each zoom level, the image tiles in a region have a coordinate domain ($0…2^{level\_zoom}$-1, $0…2^{level\_zoom}$-1). To download a satellite image (in the form of a set of tiles) from Google Map, it is necessary to calculate the tile coordinates of the tiles to be downloaded.

The zoom level (*lz*) is 0 to 19. At $lz = 0$, one tile contains the entire world image, so the map image only consists of 1 tile [14].

In general, the number of tiles at $lz = n$ is:

$$nTiles = 2^n \times 2^n \text{ tiles} = 2^{2n} \text{ tiles} \quad (6)$$

For example, at $lz = 2$, the world map consists of 2^4 tiles or 16 tiles, if $lz = 16$, the world map consists of 2^{32} tiles or 4,294,967,296 tiles (approximately 4.295 million tiles).

If there is a point *P(lon_des, lat_rad)* where *lon_des* is longitude in degrees and *lat_rad* is longitude in radial units, then the tile coordinates in the row (*xtile*) and column (*ytile*) can be calculated using the following formulas:

$$n = 2^{lz} \quad (7)$$

$$xtile = n \text{ x } ((lon\_des + 180) / 360) \quad (8)$$

$$ytile = n \text{ x } (1 - (\log_e(\tan(lat\_rad) + \sec(lat\_rad)) / \pi)) / 2 \quad (9)$$

To download a map consisting of tiles representing the urban village area, the coordinates of the outer village boundaries, namely T_1, T_2, T_3 and T_4 need to be changed to tile coordinates. At each coordinate $T_i(long\_T_i, lat\_T_i)$, the *x_tile* and *y_tile* coordinates are calculated using the formula below:

If $n = 2^{lz}$ and t = tile size = 256, then:

$$x\_tile_i = (((t/2) + long\_T_i) \text{ x } (t/360)) \text{ x } n \quad (10)$$

$$y\_tile_i = ((t/2) + 0.5 \text{ x } \log((1+sin\_y)/(1-sin\_y))) \text{ x } (- (t/(2\pi)) \text{ x } n \quad (11)$$

where

$$sin\_y = \sin(lat\_T_i \text{ x } \pi/180) \quad (12)$$

(4) Download a quadrilateral satellite image based on tile coordinates: Tile coordinates *T1* (x_tile_1, y_tile_1), *T2* (x_tile_2, y_tile_2), *T3* (x_tile_3, y_tile_3) and *T4* (x_tile_4, y_tile_4) and then used to calculate the number of tiles on the horizontal side and on the vertical side with the following formula:

Number of horizontal tiles:

$$nTile\_x = T2_{x\_tile2} - T3_{x\_tile3} + 1 \quad (13)$$

Number of vertical tiles:

$$nTile\_y = T4_{y\_tile4} - T1_{y\_tile1} + 1 \quad (14)$$

Width of the map (to be downloaded), $x\_map = t \text{ x } nTile\_x$ and the map length uh, $y\_map = t \text{ x } nTile\_y$, where t = width or length in pixel unit = 256.

Using the tile coordinate parameters in the Google Map API, each tile containing part of the village map is then downloaded. For example, the downloaded image of the Sukamaju village is shown in Figure 8. The downloaded tiles are then combined into a rectangular image.

Figure 8. *Six tiles of village of Sukamaju, district of Cibeunying Kidul.*

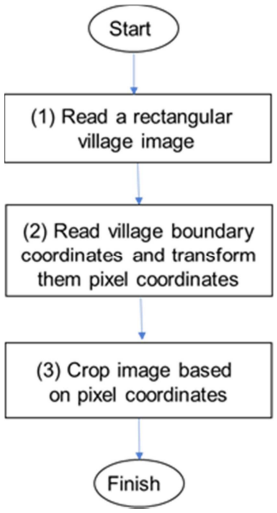

Figure 9. *Stages of preparing village image data.*

The steps of cutting or croping the village square image is depicted on Figure 9. After the quadrangle satellite image of the village is read, step 2 and 3 are explained as follows:

Step (2) Change each geographical coordinate of the urban village boundary to pixel coordinates on the map image: As the downloaded map is rectangular, while the village has certain irregular boundaries, the downloaded map must be "cut" based on the decimal coordinates of the village boundary of GeoJSON format. For this purpose, these coordinates need to be converted to pixel coordinates.

If $long_{min}$ and $long_{max}$ represent the minimum and maximum longitude values, lat_{min} and lat_{max} represent the minimum and maximum latitude values on the downloaded map, then the x and y limits in degrees can be calculated by:

$$x_{left} = long_{min} - (long_{min} \bmod oneDegree) \quad (15)$$

$$x_{right} = (long_{max} - (long_{max} \bmod oneDegree)) + oneDegree \quad (16)$$

$$y_{top} = -1 \times (|lat_{max}| - (|lat_{max}| \bmod oneDegree)) \quad (17)$$

$$y_{bottom} = -1 \times (|lat_{min}| - (|lat_{min}| \bmod oneDegree)) - oneDegree \quad (18)$$

with $oneDegree = 360/(2^{lz})$ where lz = zoom level.

If the map coordinates denoting the boundaries of a village are defined as $boundaries = \{(long_1, lat_1), (long_2, lat_2),..., (long_n\ lat_n)\}$, then the downloaded image has p = length of the downloaded image and l = width in pixels, then the pixel coordinates of the boundaries of a district will be calculated pixel-boundaries = $\{(xb_1, yb_1), (xb_2, yb_2),..., (xb_n, yb_n)\}$.

The i^{th} pixel-boundaries,(xb_i, yb_i) is computed as follows:

$$xb_i = \frac{l \times (|long_i| - |x_{left}|}{|x_{right}| - |x_{left}|} \quad (19)$$

$$yb_i = \frac{p \times (|lat_i| - |y_{bottom}|}{|y_{top}| - |y_{bottom}|} \quad (20)$$

Step (3) Cut the downloaded image according to the village boundaries: The results calculated in step (5), *pixel-boundaries*, are then used to create a mask on the image. In this mask, the pixels inside the village area are 1 while those outsides are 0. For example, Figure 10a shows the mask for the Sukamaju village. To get a village image, for each pair of the same pixel coordinates, each pixel on the mask is then multiplied by the pixels in the downloaded rectangular image. Figure 10b shows the image for the Sukamaju village which is the result of cutting a rectangular image with village boundaries.

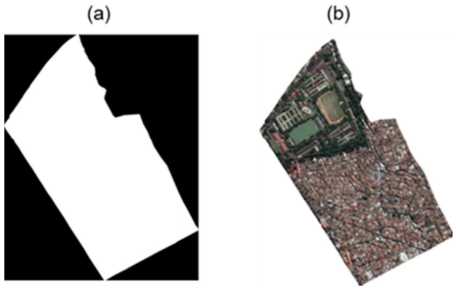

Figure 10. *Example of: (a) mask image, (b) village image as the result of image cutting.*

Stage-2: Detecting, mapping, and computing green area of the village

After the village image is prepared (cropped), the following steps are carried out:

(1) Transform the RGB image into a 3-dimensional matrix so that it can be fed to the k-Means algorithm;
(2) Conducting experiments to find the best number of clusters;
(3) Clustering of image pixels. In the results of the clustering of image pixels, a cluster that represents the tree cluster is selected, the rest is not taken into account (the pixels are changed to one color, for example gray);
(4) Calculate the green area for each urban village area.

The discussion of each stage is given below.

(1) Transformation of RGB Image to 3-Dimensional Matrix: As described in Sub-section 3.3, the k-Means clustering algorithm accepts input in the form of a matrix with dimensions (m, n), where the elements are numeric values.

The cropped satellite image for the village area is a RGB (red, green, blue) digital image, so it has dimensions (l, p, 3), where l represents the width and p represents the length of the image. Therefore, the RGB image needs to be transformed into an array with dimensions (l x p, 3).

If: $R(0..m\text{-}1,\ 0..n\text{-}1)$, $G(0..m\text{-}1,\ 0..n\text{-}1)$, $B(0..m\text{-}1,\ 0..n\text{-}1)$ and the transformation result is $A(m$ x $n,3)$, then:

$A(0,0) = R(0,0)$, $A(0,1) = G(0,0)$, $A(0,2) = B(0,0)$

$A(1,0) = R(0,1)$, $A(1,1) = G(1,1)$, $A(1,2) = B(0,1)$

...

$A(m\text{-}1$ x $n\text{-}1,0) = R(m\text{-}1,\ n\text{-}1)$, $A(m\text{-}1$ x $n\text{-}1,1) = G(m\text{-}1,\ n\text{-}1)$, $A(m\text{-}1$ x $n\text{-}1,2) = B(m\text{-}1,\ n\text{-}1)$

An illustration of the transformation for a digital image with dimensions (4,3) to an array (12,3) is given in Figure 11.

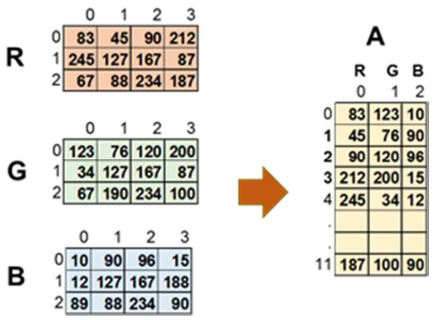

Figure 11. *Transformation of an RGB image with dimensions (4,3) to an array A(12,3).*

(2) Experiments for finding the best cluster number

The results of the transformation in stage 1 are then fed to the k-Means algorithm with the illustration given in Figure 12. The experiments were performed with a value of k = 2 to 14 with the aim of finding the best k value.

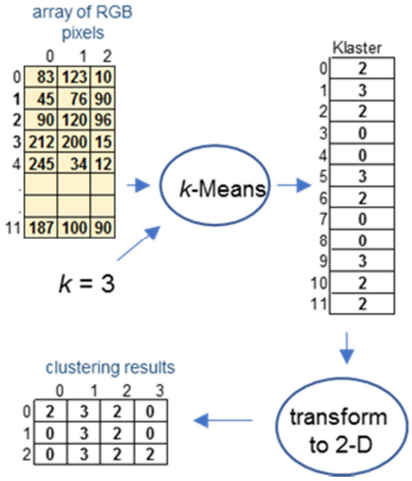

Figure 12. *Illustration of clustering an image with k = 3.*

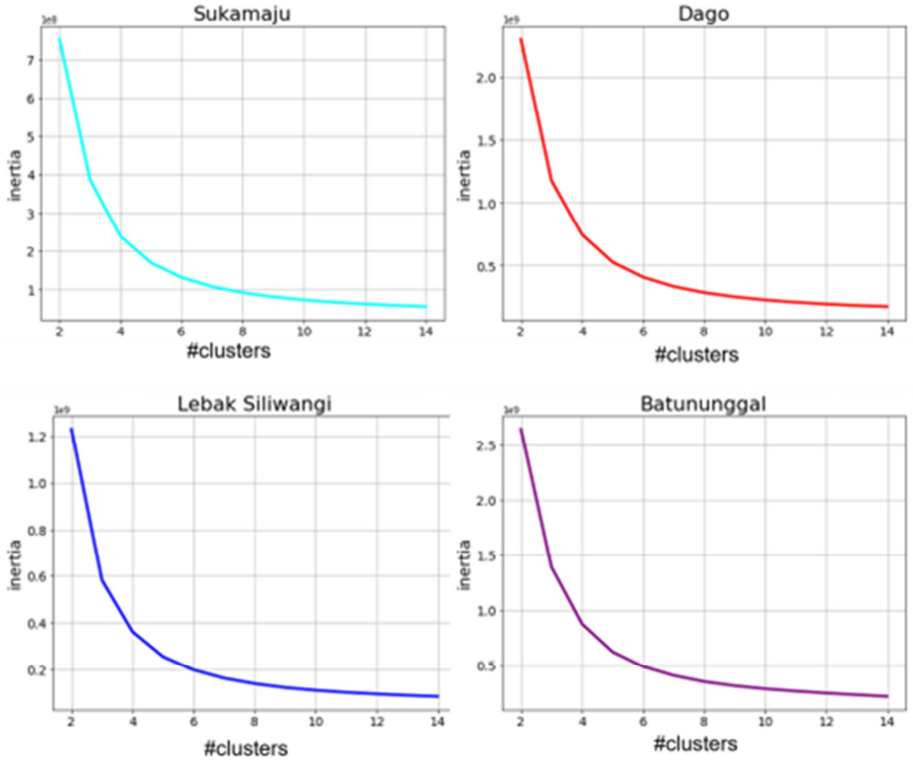

Figure 13. *Graph of inertia vs. the number of clusters of 4 village images.*

In Figure 13, it is shown that the elbow (of the cluster number) is 5. Experiments were then carried out for clustering with k = 5, 6 and 7, and the results were observed. For trees detection and calculation of green area, it was found that the best one is at $k = 5$.

So, the clustering of all village images of Bandung is done with the number of clusters (k) 5.

Figure 14. *Visualization of the original image (left) and the result of detected green areas (right) for the image of the Lebak Siliwangi village.*

(3) Clustering of village image pixels

Next, the cluster that represents the tree color (green) is selected, while the others are ignored. The selection of clusters that represent trees is carried out experimentally with visual evaluation (comparing the original image vs the image that has been colored green in the area that represents the tree area). An example of this visualization is given in Figure 14.

(4) Computing the green area

The steps are as follows:

a) The area of each pixel is calculated, L_{piksel} (m^2); the result is $L_{piksel} \sim 5.35$ m^2, which is enough to cover 1 tree.

b) The number of green pixels per village, N_{piksel}, is calculated.

c) The green area, LAH, is calculated using the formula $LAH = N_{piksel} \times L_{piksel}$ (m^2).

An illustration of LAH calculation is given in Figure 15.

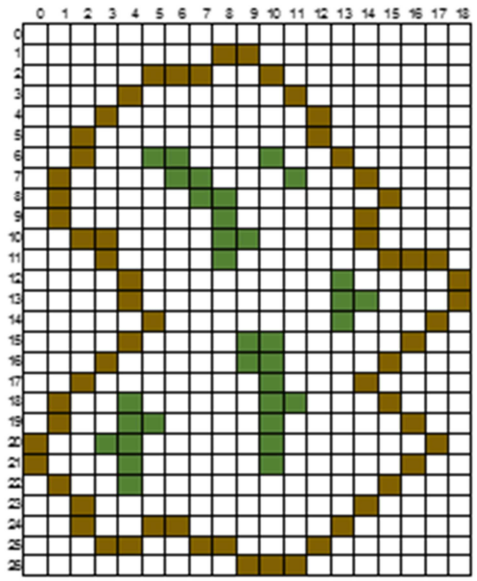

Figure 15. *Illustration of a village image that has 33 green pixels representing the green area.*

4. Results and Discussions

The techniques or methods discussed in Section 3 are implemented using the Python programming language. Satellite images were downloaded using the Google Map API, village images data are prepared using various Python libraries (PIL, NumPy, Pandas, SciPy, etc.), while clustering the prepared images with the k-Means algorithm and its evaluation was carried out using the Scikit-learn library. The results of the experiments are presented in the two sections below.

4.1. Satellite Images of Villages

On the website page of http://data.bandung.go.id, there are 149 GeoJSON files containing the coordinates of the boundaries of 149 villages in the city of Bandung. All of the files have been downloaded, were used to determine the coordinates of the image tiles from Google. All of the image tiles have been downloaded and then used to construct the rectangular image of every village.

Table 1 shows number of tiles for 5 villages. In general, the larger the area, the more tiles need to be downloaded. However, it also depends on the boundaries or shape of the village.

Table 1. *The number of tiles for five villages.*

No	Village	District	Area (ha)	Tile Width	Tile Height	# Tiles
1	Batunung-gal	Bandung Kidul	183	5	4	20
2	Ciumbuleuit	Cidadap	489	7	6	42
3	Ledeng	Cidadap	153	3	6	18
4	Dago	Coblong	266	4	5	20
5	Lebak Siliwangi	Coblong	111	3	3	9

4.2. The Results of Data Preparation, Trees Detection and Green Areas

All of the 149 RGB village rectangle images have been cropped and clustered into 5 clusters using *k*-Means. Then, one cluster that represent trees are selected. Example of comparison between the original image and the results of detected green area for 4 villages is given in Figure 16.

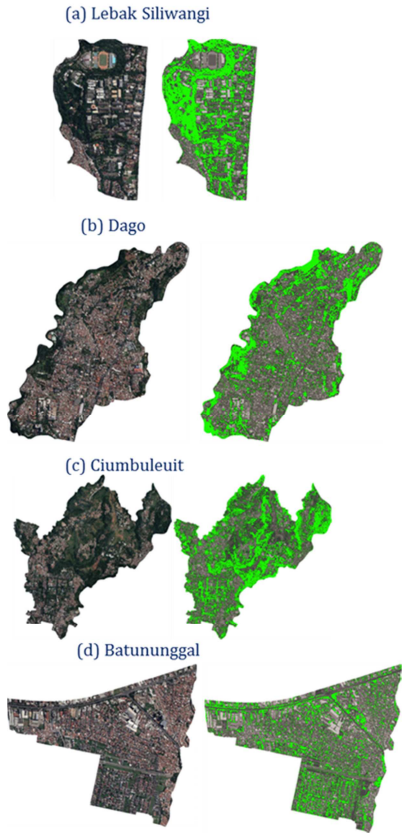

Figure 16. *Original image (left) and the detected green area (right) for 4 villages.*

By comparing the original image and the clustering results marked with green pixels, it is found that *k*-Means [3, 9, 12] with $k = 5$ (found by the Elbow method [15]) effectively cluster the image pixels into 5 clusters, where one cluster represent a group of trees and the other 4 clusters represent non-tree. By examining the clustering results, the non-tree clusters can be interpreted as buildings with dark and light colors, roads, grass or buildings with light green color. However, those 4 clusters are not regarded, as we only pick one cluster that represents trees. The clustering results show that by processing the input of array of RGB color, *k*-Means is able to place pixels representing trees (with dark green color) in one cluster based on the Euclidean distance used in the experiments.

The green area for each village is then calculated using pixels that represent trees. Example of the green area and its percentage for 10 villages is given in Table 2, while comparison among villages in 4 districts is given in Figure 17. The picture shows that the green area in the district is uneven. There are villages with green areas that are already good (above 20%), but many are still poor (barren). For instance, the percentage of green area in Cihaur Geulis village in Cibeunying Kaler district is only 5.1%. The average percentage of green area for all village is 20.8%.

Table 2. *Area and percentage of green area in 10 villages.*

No	District	Village	Area (ha)	Green (ha)	% of Green Area
1	Andir	Ciroyom	64	10.36	16.2%
2	Antapani	Antapani Kidul	117	29.61	25.3%
3	Arcamanik	Cisaranten Kulon	213	26.17	12.3%
4	Arcamanik	Sukamiskin	265	17.82	6.7%
5	Astana Anyar	Karanganyar	41	6.9	16.8%
6	Bandung Kidul	Batununggal	183	35.87	19.6%
7	Bandung Wetan	Citarum	130	40.68	31.3%
8	Buahbatu	Margasari	256	42.91	16.8%
9	Cibeunying Kaler	Cigadung	233	55.02	23.6%
10	Cibeunying Kaler	Cihaur Geulis	66	3.37	5.1%

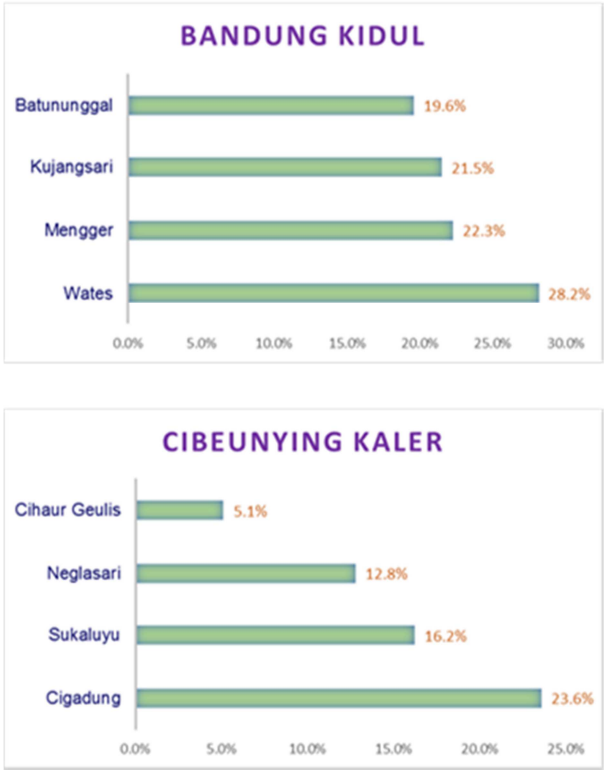

Figure 17. *Comparison of villages' green area percentage in 2 districts.*

The green area and its percentage are also calculated for each district and the results (in descending order) are given in Table 3. It can be seen that the percentages of green area vary, with some is under 20%. The greenest district is Cidadap with 31.7% of green area.

Table 3. *Green area and the percentage of 11 districts in Bandung city.*

No	District	Area (ha)	Green Area (ha)	Percentage of Green Area
1	Cidadap	842	267.21	31.7%
2	Bandung Wetan	344	93.31	27.1%
3	Coblong	731	197.04	27.0%
4	Sumur Bandung	349	88.34	25.3%
5	Bandung Kidul	542	119.74	22.1%
6	Sukajadi	528	115.21	21.8%
7	Buahbatu	746	145.29	19.5%
8	Batununggal	482	87.69	18.2%
9	Cibeunying Kaler	464	82.07	17.7%
10	Bojongloa Kaler	312	47.88	15.3%
11	Arcamanik	759	102.88	13.6%
	....	...	...	...
	Total (Bandung)	16,791	3,679	21.9%

Based on the percentages depicted on Tables 2 and 3, if the green areas are not sufficient, then strategic programs, either at the village or district level, can be designed to expand the green areas.

5. Conclusion and Recommendations

Green area of trees in each urban village can be detected and mapped from RGB satellite images using an image segmentation approach and based on pixel colors. The approximate area along with the percentage of green area for each village, district and city can also be calculated. The findings have the potency to be used by local authorities to evaluate the adequacy of green space in the lowest until upmost level of city administrative areas. The results of the evaluation can then be used as the basis for designing programs for maintaining or developing green area. The proposed method can also be used in the future (for instance, next year) when the satellite images have been updated by Google.

Village boundaries in GeoJSON format can be utilized to prepare or crop village images with masking technique. The *k*-Means clustering algorithm (with $k = 5$) is quite effective for clustering satellite images based on their pixel color such that the pixels representing trees are successfully detected, which are then used to compute area approximation of trees coverage (in hectare unit).

For further research, the trees detection can be experimented using other clustering algorithms (BIRCH, DBSCAN, etc.), the process and results are then compared to find the most optimal one. If more accurate model is needed, classification technique based on deep learning (such as discussed in [11]) can be adopted. Other research direction: In order to process wider area (provincial and country level), data collection, preparation and analysis should be carried out in a distributed system suitable for big data. Hadoop technology [17] which provides storage spread over thousands of computers can be used to manage big data of village satellite images. The MapReduce parallel computing framework on Hadoop has the potential to be used to prepare village images (for clustering purpose). Furthermore, the Machine Learning library on Spark technology [4, 8] can be used to segment (with parallel computing) the big data of prepared village images stored in the Hadoop system. For this purpose, Spark can be run on top of Hadoop by leveraging YARN (resource management and scheduling of parallel tasks) on Hadoop. Research can also be continued to detect each type of land use, for example: settlements, gardens (agricultural areas) with certain crops, vacant land, etc. The results can be used to monitor land usage in certain areas from time to time.

Acknowledgements

We would like to thank to LPPM UNPAR for its research support under contract number III/LPPM/2022-07/116-P.

References

[1] Adhzima, F., Arkeman, Y., Hermadi, I. (2022). The Clustering Rice Plant Diseases Using Fuzzy C-Means and Genetic Algorithm. Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), Vol. 6 No. 2, pp. 240 – 245.

[2] Adegun, A. A., Akande, N. O., Ogundokun, R. O. & Asani, E. O. (2018). Image Segmentation and Classification of Large-Scale Satellite Imagery for Land Use: A Review of The State of The Arts. International Journal of Civil Engineering and Technology (IJCIET), Vol. 9, Issue 11, pp. 1534–1541.

[3] Burney, S. M. A. & Tariq, H. (2014). K-Means Cluster Analysis for Image Segmentation. International Journal of Computer Applications, Vol. 96, No. 4.

[4] Chambers, B. & Zaharia, M. (2018). Spark: The Definitive Guide, Big Data Processing Made Simple. O'Reilly Media, Inc., USA.

[5] Favorskayaa, M. N., Zotina, A. G. (2021). Semantic segmentation of multispectral satellite images for land use analysis based on embedded information. Proc. of 25th Intl. Conf. on Knowledge-Based and Intelligent Information & Engineering Sytems (KES). Procedia Computer Science 192.

[6] Han, J., Pei, J., & Kamber, M. (2012). Data Mining: Concepts and Techniques 3rd Ed. The Morgan Kaufmann Series in Data Management Systems, Elsevier Inc, USA.

[7] Jeevitha, K., Iyswariya, A., Ram Kumar, V., Basha, S. M., Kumar, V. P. (2020). A Review on Various Segmentation Techniques in Image Processsing. European Journal of Molecular & Clinical Medicine, Vol. 7, Issue 4.

[8] Karau, H. and Warren, R. (2017). High Performance Spark. O'Reilly Media, Inc., USA.

[9] Kumar, J. M., Nanda, R., Rath, R. K., Rao, G. T. (2020). Image Segmentation using K-means Clustering. International Journal of Advanced Science and Technology, Vol. 29, No. 6s, pp. 3700 – 3704.

[10] Map and Tile Coordinates, https://developers.google.com/maps/documentation/javascript/coordinates, (Accessed: 20 Feb. 2022).

[11] Onyango, L. A., Waititu, A. G., Mageto, T., Kilai, M. (2022). A Hybrid Classification Model of Artificial Neural Network and Non Linear Kernel Support Vector Machine. International Journal of Data Science and Analysis. Vol. 8, No. 2, pp. 47-58. doi: 10.11648/j.ijdsa.20220802.15.

[12] Panwar, P., Gopal, G., Kumar, R. (2016). Image Segmentation using K-means clustering and Thresholding. International Research Journal of Engineering and Technology (IRJET), Vol. 03, Issue 05.

[13] Pertiwi, A. P., Roth, A., Schaffhauser, T., Bhola, P. K., Reuß, F., Stettner, S., Kuenzer, C., Disse, M. (2021). Monitoring the Spring Flood in Lena Delta with Hydrodynamic Modeling Based on SAR Satellite Products. Remote Sensing, 13, 4695. https://doi.org/10.3390/rs13224695

[14] Slippy map tilenames, https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames#ECMAScript_.28JavaScript.2FActionScript.2C_etc..29 (accessed: 27 Feb 2022).

[15] Syakur, M. A., Khotimah, B. K., Rochman, E. M. S., & Satoto, B (2018). D. Integration k-means clustering method and elbow method for identification of the best customer profile cluster. IOP Conference Series: Materials Science and Engineering, 336, 012017.

[16] Tin, S. N. and Muttitanon, W. (2021). Analysis of Enhanced Built-up and Bare Land Index (EBBI) in the Urban Area of Yangon, Myanmar. International Journal of Geoinformatics, Vol. 17, No. 4, Pp. 85-96. https://doi.org/10.52939/ijg.v17i4.1957.

[17] White, T. (2012), Hadoop: The Definitive Guide 3rd Ed., O'Reilly Media, Inc., USA.

[18] Wijayanti, R. F., Jaelani, L. M., Handayani, H. H. and Chu, H. J. (2021). Drought Index Mapping in Java Island Using Sentinel-3 SLSTR, International Journal of Geoinformatics, Vol. 17, No. 4, Pp. 97-108. https://doi.org/10.52939/ijg.v17i4.1959.

[19] Yang, N. & Tang, H. (2021). Semantic Segmentation of Satellite Images: A Deep Learning Approach Integrated with Geospatial Hash Codes. Remote Sensing, Vol. 13, No. 2723. https://doi.org/10.3390/rs13142723