Handling Massive Data Size Issue in Building Footprint Extraction from high resolution satellite images

Sohaib K.M. Abujayyab[1], Ismail. R. Karas[2] S. Sahin[3], M. Topal[4] and E.N. Kalkar[5]

Dept. of Computer Engineering, Karabuk University, 78050 Karabuk, Turkey
[1] s.jayyab@hotmail.com
[2] irkaras@gmail.com
[3] sergensahin96@gmail.com
[4] mehmettopalofficial@gmail.com
[5] eminenurkalkar@gmail.com

Abstract. Building information modeling BIM relaying on plenty of geospatial information such as building footprints. Collecting and updating BIM information are real challenge. Building footprints were automatically extracted from high resolution satellite images utilizing machine learning algorithms. Constructing required training datasets for machine learning algorithms and testing data is computationally intensive. When the analysis performs in large geographic areas researcher struggling from out of memory problems. The requirement of developing improved, less expensive methods for accomplishing this computation is urgent. This paper targeting to handling massive data size issue in building footprints extraction from high resolution satellite images. This article developed a method to process the spatial raster data based on the chunks computing. Chunk-based decomposition raster array to several tiny cubes. Cubes supposed to be small enough to fit into available memory and prevent memory overflow. The algorithm of method developed using Python programming language. Spatial data and developed tool were prepared and processed in ArcGIS software. Matlab software utilized for machine learning. Neural networks implemented for extracting building footprints from satellite images with 99.3% performance accuracy for testing dataset. To demonstrate the performance of our approach, satellite image having (1409 columns, 1346 rows, cell size 0.61 meter, area 0.7056928594 km2, 7586056 pixels in 4 bands) feed to the tool. The image impossible to be totally handle in central processing unit CPU. The image divided to 36 chunks using 250 * 250 rows and columns. Full analysis spent 1 hour. Selecting the chunks size is critical issue, if the chunks size selects not properly, the chunks process will be useless. The employed method shows that chunk computing can solve the memory overflow in personal computers when processing large files. Consequences demonstration that image smoothly processed, migrated and building footprints extracted effetely, avoiding memory overflow. The lessons learned from the tests are summarized. Furthermore, future research should include global and focal raster computation beside local raster operations.

Keywords: Building Information Modelling, Building Footprint Extraction, Massive Data Size, High Resolution Satellite Images, Neural Networks.

1 Introduction

Building Information Modelling (BIM) is a digital representation of physical and functional attribute of buildings [1]. A BIM is a mutual knowledge resource for data about a buildings establishing a reliable foundation for decisions during its life-cycle [2–4]. A BIM is an intelligent 3D model-based process that gives engineering, architecture, and construction professionals the insight and tools to more efficiently plan, design, construct, and manage buildings and infrastructure [4].

Apparently, key stage in BIM system is collecting and storing digital information of every aspect of the existing buildings. Digital information about the buildings help decision makers to optimize their actions. Optimized actions can be affected from the design until construction and managing stage during buildings life-cycle [5]. In addition, updating the information issue is a challenging process in BIM system. Updating the information is time consuming, financially costly and time challenging. High resolution satellite images (HRSI) became a great solution for collecting and updating BIM spatial data. Application areas of HRSI have been substantially increased due to availability of sub-meter spatial high-resolution of satellite images such as, QUICKBIRD, WorldView and IKONOS [6–8]. By processing HRSI, several information can be extracted automatically such as detect the buildings, buildings highest, buildings footprint and roof type using machine learning algorithms. Building footprint is the area on a project site that is used by the building structure and is defined by the perimeter of the building plan. Parking lots, landscapes, and other non-building facilities are not included in the building footprints [9–11]. In the past two decades, building detection and reconstruction from remotely sensed data has been an active research topic in the photogrammetric and remote sensing communities [12,13].

Automatic extraction of building footprints from HRSI has been varied difficulties [14]. One of the main challenges is handling massive data size issue in HRSI. Issue arising from the truth that machine learning algorithms need specific data structure to apply the analysis. This data structure of machine learning algorithms is an input and target sets. Input data set consisting the parameters of each sample. The samples in building footprint extraction application area represent the pixel, while parameters represent the bands of the satellite images. So, the process to reshaping the raster spatial data to be suitable for machine learning algorithms data structure is time consuming. In addition, reshaping the spatial data and establishing the data set required massive random memory, especially if the geographical area is big, and analysis area have giant number of rows and columns due to HRSI. In addition, the increase in resolution of raster datasets has led to larger and larger data sizes [15]. Presently, datasets are on the order of gigabytes and increasing, with billions of raster cells. While computing power of the processors and size of the memory in computers have increased appreciably, legacy equipment and algorithms suited to manipulating small rasters with coarser resolution make processing these improved data sources costly.

Massive data size issue caused plenty of problems for the specialists such as out of core computation or out of memory [16]. HRSI need huge memory to process and store them. This problem totally stopes the specialists from performing the analysis or limited their analysis only in tiny geographical areas. In some cases, massive data size issue lead to crash the systems especially if they are using only serial processing or sequential processing in classical personal computers PC's.

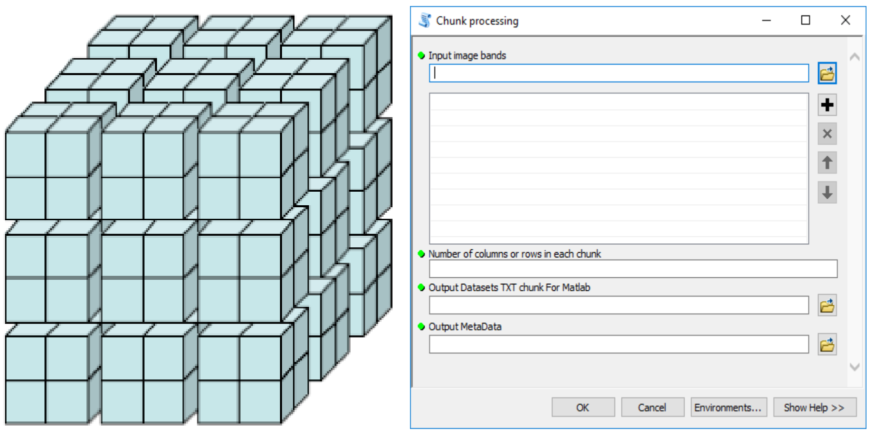
However, massive data size issue need to manage the memory carefully to avoid any crash during the Geo-processing [17]. Several methods were proposed in the past to handle this issue such as parallel processing, multitasking, distributed systems, supercomputing, cloud computing and graphics processing unit (GPU). But some these methods are not suitable for machine learning algorithms. In case of working with some environments such as ArcGIS, methods are available for the built in functions, while it is not available for the developed functions or other processing environments. Additionally, in case of the need to migrate the data to other environments for instance Matlab software, these methods are not available. In some cases, the main problem is not how long the processing is taking time, but the problem that the system is crash or stop the processing. The system stops the processing due to that the size of data need random memory more than the available memory in the device.

Thus, the main aim of this article is to handling massive data size issue in building footprint extraction from HRSI. This paper developed a method to process the spatial raster data based on the chunks computing for sequential processing. Then, the developed method implemented and tested based on cased study data. HRSI with massive data size were processed using this tool. The tool developed using Python programing language. While the machine learning algorithm implemented using Matlab software. HRSI were processed in ArcGIS.

2 Methods

The developed method designed using Python programing language. The cod considered that spatial raster data will be processes in ArcGIS and Matlab. Previously, HRSI were completely loaded to the computer, which faced the out of memory problem [18]. The developed method relay on dividing raster array to several tiny cubes, which called chunks (Fig. 1 (A)) [17]. Cubes supposed to be small enough to fit into available memory. Once the data is decomposed appropriately, each data 'chunk' can be operated on independently on by a process. At this stage, raster data are loaded into the memory and computation performed based on sequential processing cubes-by-cubes. Thus, reducing the memory usage during the Geo-computation and avoiding the crashing of the analysis.

Small cubes data structure is very suitable for machine learning algorithms data structure. First dimension of the cube represents the number of columns, second dimension represent numbers of rows and third dimension represent the band of satellite image. Since the HRSI is multispectral images and having limited number of bands (between 3 to 12), the method dividing only the first dimension and storing full third dimension. Cubes of raster data can easily reshape to the suitable and required machine learning datasets. Every cube will represent one separated dataset and will be migrated to Matlab and be tested individually. By this process, the model shortens the image processes through reading the pixel values directly from raster structure. Previously, models were convert the raster data to tabular data based on sampling strategy. Sampling strategy is convert the raster to points and extract the pixel values and store it in the linked attribute data with the grid of points. Then, migrate these data to machine learning software. This previous process is highly time consuming compared with the developed method. Instead of reading points, this method read a small piece from the image, make some calculations, write the output on the disk, and release the memory before repeating the process with the next piece.



(A) (B)

Fig. 1. (A) HRSI 3D array divided to chunks. Figure showing how storing a three dimensional chunked storage structure. the chunk size is (longitude) x (latitude) x (band). (B) Developed tool in ArcToolbox using Python for chunk processing.

In addition, chunks data structure is suitable for parallel computing and easily to be configured. Furthermore, the analyzer of HRSI can know and control number of chunks based on their available memory and devises. This method is simple and more acceptable for remote sensing society. The method code developed to be suitable for ArcGIS-Arc toolbox environment. One toolbox developed to apply the method to raster data. The tool illustrated in (Fig. 1 (B)). The first input of the tool is the HRSI's. the user need to added the images band by band. The input design to be raster layer, which accept bands from table of content or from the hard drive directly without the need to add the data to table of content. Adding the data to table of content is memory consuming. Second input is chunk size. Third input is the output workspace for the TXT datasets. Finally, forth input/output is the location of each chunk metadata. Metadata file help to stitched back chunks together via mosaicking it to one meaningful output image.

Chunks size parameter must be defining carefully. defining chunks size is based on the experiments. If chunks size is too small, queueing up operations will be extremely slow, because each chunk has storage cost. Conversely, if chunks size is too big, benefits of chunks computation may be wasted, because chunk array will not fit into the available memory. Thus, selecting suitable chunks size will avoid any risk for the system.

In order to well describe the developed method, code of the method is described as following;

```
# importing the required Python libraries
import arcpy
import numpy as np
import os
# Reading the input raster
inputRasters = arcpy.GetParameterAsText(0)
#defining size of chunk
chunkssize= arcpy.GetParameterAsText(1)
#defining out the Workspace
OutputMetaData = arcpy.GetParameterAsText(2)
Output= arcpy.GetParameterAsText(3)
# Loop over raster data chunks
for x in range(0, inputRasters.width, chunkssize):      # Loop over image columns
  for y in range(0, inputRasters.height, chunkssize):   # Loop over image rows
    for in_raster in inputRasters:                      # Loop over input bands
      IR = arcpy.Raster(in_raster)                      # Reading the input raster
      # Getting chunk dimensions
      mx = IR.extent.XMin + x * IR.meanCellWidth  # defining minim X coordinate
      my = IR.extent.YMin + y * IR.meanCellHeight # defining minim Y coordinate
      lx = min([x + chunkssize, IR.width])        # defining maximum X coordinate
      ly = min([y + chunkssize, IR.height])       # defining maximum Y coordinate
      # Extract chunk data
      Chunk = arcpy.RasterToNumPyArray(IR, arcpy.Point(mx, my), lx-x, ly-y)
      V = np.ravel(Chunk)
      RL1.append(V)
      NewNpArray = np.array(RL1)
            RL = np.transpose(NewNpArray)
    raster += 1
    BlockChank = os.path.basename(in_raster)
    BlockChank2 = str(BlockChank) + "_" + str(raster)
    Output3 = str(Output) + str(BlockChank2) + ".txt"
    np.savetxt(Output3, RL, delimiter=" ", fmt="%s") # Storing the dataset of chunk
    Arrayshape = Chunk.shape
    COLUC = Arrayshape[0]
    ROWC = Arrayshape[1]
    MCW = IR.meanCellWidth
    MCH = IR.meanCellHeight
    Meta = str(MCW) + " " + str(MCH) + " " + str(mx) + " " + str(my) + " " + str(ROWC)
         + " " + str(COLUC)
    MetaData.append(Meta)
np.savetxt(OutputMetaData, MetaData, delimiter=" ", fmt="%s") # Storing the Metadata
```

The developed method and tool employed through a case study data. HRSI obtained and implemented (Fig. 2). The obtained satellite image is high resolution image. The image consisting 1409 columns, and 1346 rows. The cell size of the image is (0.61 meter, 0.61 meter). Total area of case study is 0.7056928594 km^2. The image with 1896514 pixels in one band and 7586056 pixels in 4 bands is impossible to be totally handle in central processing unit (CPU). This image need advanced processor and big memory to be completely processed by one set. In a case the analyzer need to perform augmentation process and increase the number of input images, the difficulties and problem will be increased.

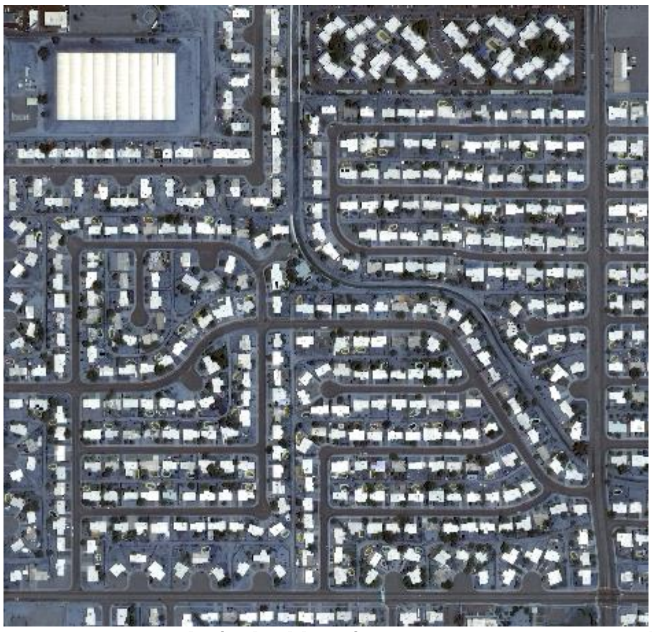


Fig. 2. HRSI as case study for building footprints extraction using chunks processing. The image consisting 7586056 pixels in 4 bands.

By reading the corresponding numbers of rows, columns and bands of the image, 250 columns and 250 rows were defined as the chunks size. That's mean that every chunk dataset consisting 62500 pixels. The script converts a multiband raster to a three-dimensional NumPy array. The array automatically divided to data chunks. Then TXT datasets files were stored and migrated to Matlab software. The TXT format make analysis flexible to be implement in different software's. In Matlab, datasets were trained and tested through neural network [16,19–29]. Neural networks are a collection of functions, trying to simulate the biological cell work of human, which developed for patterns recognition. The impression of Neural networks is according to the belief that working of human brain, by making the right connections, can be copied using silicon and wires as living neurons and dendrites [30].

The human brain is consisting of 86 billion nerve cells called neurons. Neurons are associated to further thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. Neurons inputs generate electric impulses, which rapidly moved over the neural network. The neuron can take input data and execute simple operations on the data. The outcome of these processes is passed to other neurons. The output at each neuron is called its activation or node value. Each connection is linked with weight. Neural networks are able to training, which takes place by altering weight values. Neural networks can understand sensory data through a kind of machine perception, clustering or labeling raw input. The recognized patterns are numerical, stored in vectors as parameters. Most of application areas their data need to be reshaped to the suitable data structure such as sound, images, text and time series data. Neural network applied for building footprint extraction from HRSI. Neural networks applied in Matlab software. Backpropagation neural network utilized. Sample points of training data set was consisting 392 points. an example extract from the case study data illustrated in Table 1. The table show the explanatory parameters data and their target values. Data set of training stage extract based on point processing, due to the storage size is limited. Neural network used 4 input bands for the input layer and utilized 33 neurons in the first hidden layer. The network reach to the optimal performance after 67 iterations if training.

Extraction process of building footprints implemented based on chunks datasets one by one. The outputs of neural network were stored and migrated to ArcGIS again. Outputs of neural networks of every chunk were converts to raster, and recombines the image via mosaicking. Thus, single band classified image of building footprints were produced.

Table 1 Example part of training dataset of sample points

Input (explanatory parameters)				**Target (building footprints)**
Band 1	**Band 2**	**Band 3**	**Band 4**	
1020	1640	1266	1252	1
993	1622	1266	1271	1
479	790	670	747	0
1053	1721	1339	1344	1
1170	1791	1352	1235	1
503	857	765	824	0
416	618	479	504	0
1221	1933	1476	1380	1
456	723	587	635	0

3 Result

Data of case study successfully tested using the developed method and tools. In this analysis, the tool divided the study area to 36 chunks based on (250 * 250 pixels). Average storage size of chunks files is 1 MB, which easily can be processed by any CPU. Full analysis spent 1 hour. The tool directly divided the image, read pixel values and reshaped every chunk array. Then arrays stored in TXT files and migrated to Matlab. In Matlab software, chunks arrays were tested one by one through looping over the files. Testing processed were applied to automatically extract building footprint from HRSI. Extraction process implemented using neural networks as machine learning algorithm. NN highly recommended for satellite image processing.

After training the model, high testing accuracy achieved based on the sample dataset. Confusion matrix utilized as accuracy evaluation metric. Final achieved performance accuracy was 99.3%. The full image was smoothly processed through looping over the 36 chunks. Then, chunks data effectively moved to ArcGIS and recombine in one full image. The image of case study processed without facing the out of memory problem. This full image represents the extracted building footprint from the high resolution satellite images. Extracted building footprint were illustrated in (Fig. 3). The advantages of optimizing raster analysis tasks with Python chunks are evident from this figure.

In the last part, the most important thing was to get memory usage reduced. In this case study, method tried to developed a suitable tool to solve memory usage problem due to huge data stuck. So, code implemented for handling massive data size issue in satellite image. The method was effectively implemented in the application area of building footprints extraction [31–33], and highly recommended for further application.

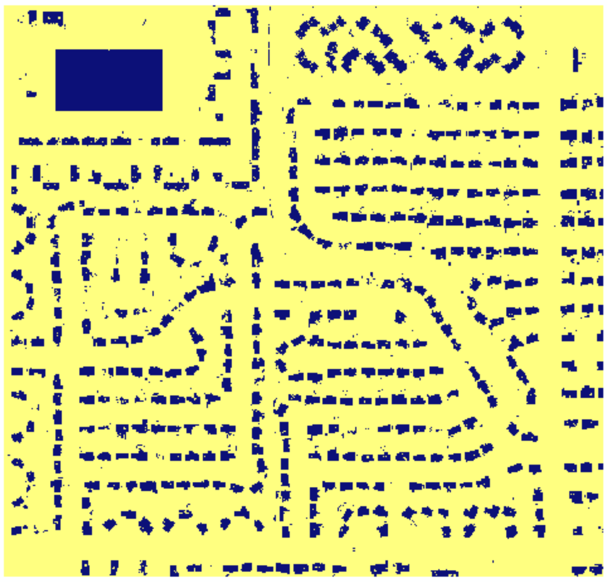


Fig. 3. Output of the neural network model based on chunk processing. The map illustrating the extracted building footprint from the high resolution satellite images

4 Conclusion

Automatically extraction of building footprints from high resolution satellite images utilizing machine learning algorithms is computationally intensive than regular raster Geo processing. Working with full images for big geographic area is challenging in central processing unit, and facing out of memory problems. In addition, the cost of migrating big data and repeating the processing to gain up to date information. Building training and testing datasets for machine learning algorithms required high memory usage. Highly usage of memory lead to stop the analysis or even to crash the analysis systems. Thus, the need of developing a methodology to solve high memory usage in spatial data is essential. developed a method to process the spatial raster data based on the chunks computing for sequential processing

This article aims to established a method to process the spatial raster data based on the chunks computing. Chunk computing breakdown raster array to several tiny cubes. Cubes must be small enough to fit into offered memory and prevent memory crash. Instead of reading and processing full high resolution satellite image, this method read a small portion from the image, apply some Geo computation, write the output on the disk, and release the memory before repeating the process with the next portion. The method function established using Python programming language. Spatial information and established toolbox processed and prepared in ArcGIS environment. Matlab environment used for machine learning. Neural networks executed for extracting building footprints from high resolution satellite images with 99.3% performance accuracy for testing dataset. Confusion matrix applied as accuracy evaluation metric. To validate the performance of proposed method, massive satellite image having (1409 columns, 1346 rows, cell size 0.61 meter, area 0.7056928594 km2, 7586056 pixels in 4 bands) feed to the toolbox. In the normal situation, the image is impossible to be totally handle in central processing unit CPU. Image separated to 36 chunks. Every chunk is (250 * 250) consisting 62500 pixels. All the analysis spent 1 hour for processing. Selecting the chunks size is critical issue, if the chunks size selects not properly, the chunks process will be useless. Outcomes proved that high resolution satellite images can be smoothly processed, migrated, building footprints extracted effetely, and avoiding out of memory problem. The employed method shows that Chunk computing can solve the memory overflow in personal computers when processing large files. The developed method is suitable to be implement in an affordable lightweight desktop environment, rather than, needing high computing capability that recommended previously. Future work can be including an expanding to this research to support several and further processing functions beside the building footprints extractions. In addition, increase the functionality of the method to automatically calculate the optimal chunk size based on different computing power. Additionally, future works should include global and focal raster computation beside Local raster operations.

Acknowledgment

This study has been supported by 2221 – Fellowship Program of TUBITAK (The Scientific and Technological Research Council of Turkey). We are indebted for their supports.

References

[1] H. Kreiner, A. Passer, H. Wallbaum, Energy Build. 109 (2015) 385–396.
[2] C. Cavalliere, G.R. Dell'Osso, A. Pierucci, F. Iannone, J. Clean. Prod. 199 (2018) 193–204.
[3] M.F. Muller, F. Esmanioto, N. Huber, E.R. Loures, O. Canciglieri, J. Clean. Prod. 223 (2019) 397–412.
[4] X. Yin, H. Liu, Y. Chen, M. Al-Hussein, Autom. Constr. 101 (2019) 72–91.
[5] Wikipedia, Wikipedia (2019).

[6] Japanese Earth observing satellite, (2019).
[7] Alaska Satellite Facility's, (2011).
[8] H.A. Nefeslioglu, B.T. San, C. Gokceoglu, T.Y. Duman, Int. J. Appl. Earth Obs. Geoinf. 14 (2012) 40–60.
[9] Y. Park, J.-M. Guldmann, Comput. Environ. Urban Syst. 75 (2019) 76–89.
[10] R. Sinha, M. Lennartsson, B. Frostell, Build. Environ. 104 (2016) 162–171.
[11] B. Green, Green, Build. (2019).
[12] O. Tournaire, M. Brédif, D. Boldo, M. Durupt, ISPRS J. Photogramm. Remote Sens. 65 (2010) 317–327.
[13] J. Huang, X. Zhang, Q. Xin, Y. Sun, P. Zhang, ISPRS J. Photogramm. Remote Sens. 151 (2019) 91–105.
[14] N.L. Gavankar, S.K. Ghosh, Eur. J. Remote Sens. 51 (2018) 182–193.
[15] M. Hamzeh, R. Ali Abbaspour, R. Davalou, Environ. Sci. Pollut. Res. 22 (2015) 12511–12524.
[16] D. Li, S. Wang, D. Li, (2015).
[17] J. Li, P.M. Finn, M. Blanco Castano, ISPRS Int. J. Geo-Information 6 (2017).
[18] M. Norman, H.Z. Mohd Shafri, M.O. Idrees, S. Mansor, B. Yusuf, Geocarto Int. (2019) 1–24.
[19] L. Xu, P. Gao, S. Cui, C. Liu, Waste Manag. 33 (2013) 1324–1331.
[20] C. Cheng, W. Niu, Z. Feng, J. Shen, K. Chau, Water 7 (2015) 4232–4246.
[21] N. Samadiani, H. Hassanpour, J. Electr. Syst. Inf. Technol. 2 (2015) 207–218.
[22] M.S. Islam, M.A. Hannan, H. Basri, A. Hussain, M. Arebey, Waste Manag. 34 (2014) 281–290.
[23] M.K. Younes, Z.M. Nopiah, N.E.A. Basri, H. Basri, M.F.M. Abushammala, M.Y. Younes, Waste Manag. (2015).
[24] K.-Y. Song, H.-J. Oh, J. Choi, I. Park, C. Lee, S. Lee, Adv. Sp. Res. 49 (2012) 978–993.
[25] Y. Fu, Y. Zhao, Y. Zhang, T. Guo, Z. He, J. Chen, Environ. Earth Sci. 68 (2013) 1495–1505.
[26] E. Garcia-Breijo, J. Atkinson, L. Gil-Sanchez, R. Masot, J. Ibañez, J. Garrigues, M. Glanc, N. Laguarda-Miro, C. Olguin, Sensors Actuators A Phys. 172 (2011) 570–582.
[27] A. Yalcin, S. Reis, A.C. Aydinoglu, T. Yomralioglu, CATENA 85 (2011) 274–287.
[28] M. Abbasi, A. El Hanandeh, Waste Manag. (2016).
[29] V. Nourani, B. Pradhan, H. Ghaffari, S. Sharifi, Nat. Hazards 71 (2014) 523–547.
[30] K. Al-Mahallawi, J. Mania, A. Hani, I. Shahrour, Environ. Earth Sci. 65 (2012) 917–928.
[31] B. Vallet, M. Pierrot-Deseilligny, D. Boldo, M. Brédif, ISPRS J. Photogramm. Remote Sens. 66 (2011) 732–742.
[32] M. Brédif, O. Tournaire, B. Vallet, N. Champion, ISPRS J. Photogramm. Remote Sens. 77 (2013) 57–65.
[33] R. Alshehhi, P.R. Marpu, W.L. Woon, M.D. Mura, ISPRS J. Photogramm. Remote Sens. 130 (2017) 139–149.