Distributed Framework for Processing High-Resolution Remote Sensing Images

T. Naga Raju, Chittineni Suneetha

Abstract:- ***Now-a-days, sensing of remote satellite data processing is a very challenging task. The current development of satellite technology has led to explosive growth in quantity as well as the quality of the High-Resolution Remote Sensing (HRRS) images. These images can sometimes be in Gigabytes and Terabytes, which is heavy to load into the memory and also takes more time for processing. To address the challenges of processing HRRS images, a distributed map Reduce framework is proposed in this paper. This paper reflects Map-reduce as a distributed model using the Hadoop framework for processing large amounts of images. To process large amounts of images, block-based and size-based methods are introduced for effective processing. From the experiments, the proposed framework has proven to be effective in performance and speed.***

Index Terms: Hadoop framework. High Resolution, Remote Sensing, Map-reduce

I. INTRODUCTION

To examine geospatial data, there is a massive need of earth observation satellites which are gradually being used as remote sensor images [1-21]. The sorts of satellites that observe earth produce very high-resolution images whose size can vary from very few MB to a range of GB. These images contain huge information if extracted in a proper way. Several sorts of information can be retrieved from the images produced by satellites such as terrain mapping, mining details, mine locations, weather monitoring, entertainment, population density, farming detail, etc. there lies a crucial phase to extract valuable and beneficial data from the images but not in collecting these images. Effectively extracting worth-full information from remote sensing and medical images [23-26] with high resolutions, highly data-intensive images is a challenging task [3].

Kothuri et al. [4] came up with an approach were to handle the image data, the oracle database is used, by which the problems of integrity and consistency of data are maintained effectively. This approach fails if the image data is huge. There are several limitations, such as hardware limitation, single node failure, inefficiency in query management. Upon advancement of Big Data Technology in Open Source, few methodologies of using Big Data technology for processing images delivered by remote sensors are proposed [4]. One such proposal is Zheng et al. [7] has proposed a Hadoop platform which is used to store vector data. It uses the Distributed File System framework to store the data; Map Reduce to analyze the data with an open-source distributed framework.

Revised Manuscript Received on October 20, 2019.

T. Nagaraju, Research Scholar Department of Computer Science & Engineering, Acharya Nagarjuna University, Guntur, (AP.), India.

Dr. Ch. Suneetha, Associate Professor, Department of Computer Applications, RVR&JC College of Engineering, Chowdavaram, (AP.), India.

The Hadoop framework mainly comprises of HDFS [5] and Hadoop Map Reduce [6]. This has effectively overcome the problematic situations such as single point of failure and non-extensibility. However, numerous small files are generated enormously in the way of storing data is not taken into consideration, which overburdens the primary node with image dumping. The [7] based on [4] has proposed a model to effectively manage massive image data and include all the small files into a single large data file with distributed key-value storage model which solves a huge number of small file problems. This increases data retrieval time where it doesn't describe every layer of metadata of the image.

Hadoop Map Reduce is a tool considered for processing huge amounts of data. In this methodology, whole data is divided into blocks. The most important task of Map Reduce is a division of the given input data into individual chunks of data. These chunks should be parallelly processed. These images which are processed are then stored in HBase[8]. This HBase is a distributed database framework methodology which is generally used where there is in need of large tables with non-sequential, real-time read and write access. In [9] the image after processing is saved in HBase, upon which it can update the required data at any moment to resolve the multi-temporal problem saved in remote sensing image data.

In this work, we have come up with a new proposal with improved HDFS- Hadoop Framework methodology, which can very efficiently process high-resolution images in quick and no time. Rest of the paper is organized as follows. The below both sections respectively brief with the literature survey associated with High-Resolution Remote Sensing Big Data and also a discussion of the core concepts of HADOOP distributed Framework. Next sections discuss the proposed work for processing HRRS big data and results, and discussions are given. Finally concludes the entire paper.

II. RELATED WORK

Kennedy et al. [10] have discussed the importance of Map-Reduce approach by labeling millions of images and applying the Nearest Neighbour algorithm to it. Shi et al., [11] also explained the same usage of Map-Reduce for CBIR[27,28] for applying nearly 400,000 images almost. Mohammed [12] explained the concept of high-performance computing and also analyzing the performance of remote sensing data using Hadoop. In this process, two algorithms are applied one for auto – contrast using six-fold to speed up the method and for sharpening an eight-fold algorithm. Temizel et al. proposed a distributed method named ad Hadoop and Map Reduce to accomplish an analysis of ballistic images, which requires a huge image database [13]. Temizel used correlation technique, which is applied for a high computational demand, which reduces processing time.

This method used a master configuration consisting of 14 computational nodes. Ma et al. [14] have come up with challenges and opportunities in Remote Sensing (RS) Big Data computing, which has deep insight on RS data-intrinsic problems, analysis of RSBD and numerous techniques for processing implementation for RSBD.

Most recently, a standard method is followed for the handling of large scale image processing techniques using Map-Reduce framework because of its peculiar features like enormous scalability, flexible programmability, competitive fault tolerance and very less cost for its deployment. On the achievement of Map-Reduce model and High-Performance Computing (HPC) skill, numerous techniques are been proposed (Ex: distributed and parallel computing based) in the literature for large scale remote sensing data sets. Later Li et al. Tried to attempt for optimizing the computational time on images produces by satellites by using Hadoop and Map Reduce. Li proposed a parallel clustering algorithm for processing large scale satellite images. The procedure initiates by clustering every pixel with its nearest cluster and calculates all centers of the clusters on the base of each and every pixel in unique cluster set [15]. Jhong [16] have come up with a proposal of yet another algorithm for clustering for processing of RS image data with the use of parallel K-means approach. According to his approach, the object with similar spectral values will be clustered together without having any formal knowledge. The parallel K – means approach was insisted by Hadoop Map Reduce approach since the algorithm is very intensive in both time and memory. Jakovits et al. [17] demonstrate the tool about large size images using Hadoop Map Reduce environment. Using this tool processing of images are in a sequential manner, but they are either in smaller or in regular size altogether, the number of images processed is 48675. This was very much similar to HIPI (Hadoop Image Processing Interface) tool. HIPI works as an image processing tools available in a distributed environment. All the above mentioned methods stated that large scale image dataset requires HDFS. Every image is totally fixed for a unique map-reduce for processing. Till date, there is none of any standard design, methodology, and solution developed for HRRS pictures using one of either Hadoop or Map Reduce.

All existing works are generally concentrated only on Hadoop Map-Reduce method for data of an image in a parallel manner. One of the key elements in processing is the optimization of HDFS that isn't being well given in the above literature. But it is necessary to recognize the necessity for optimal HDFS storage as this is one of the key aspects in options providing optimal performance for image data.

III. HADOOP-DISTRIBUTED FRAMEWORK FOR IMAGE PROCESSING

Apache Hadoop is an open-source framework for storing, processing, and analyzing huge amounts of structured as well as unstructured data [8]. For processing all types of data, a network of computers is necessary. This framework is formulated for parallel processing of data which are evenly distributed to the computing nodes, in a way that a Hadoop cluster is easily scalable horizontally. Hadoop framework constituted the following modules [9] as shown in Table-1.

Hadoop	Consists of libraries and utilities required by other modules of Hadoop. 1. To save and extract large amounts of unstructured data in Hadoop, NoSQL databases (*HBase* and *Cassandra*) can be utilized. These sorts of databases support MapReduce mechanism especially suitable for volumes of unstructured data [13]. 2. In surplus to Java, Pig is an open-source language which is designated for Hadoop. 3. Collection of numerous data from several sources and store them in Hadoop, 4. Flume framework was designed for the collection of data and transmitting them instantly by placing on different locations like huge servers, mobile devices, etc. *5.* Consider *Oozie* Library for sequential execution of requests, which is allowing users to execute several requests sequentially and also each of the requests can use the 6. *Whirr* Library is recommended for Hadoop cloud systems 7. *Mahout* library is used for data mining algorithms like clustering and regression.
HDFS	Hadoop Distributed File System (HDFS) is mainly used for storage and processing of huge scale datasets. **Fig.1** below describes the architecture of HDFS.
Hadoop YARN	This is a cluster resource-management platform. Its responsibility is to manage clusters and also for scheduling of users' applications. Map-Reduce 2.0 (MRv2) version is used.
MapReduce	Hadoop Map-Reduce is generally applicable for huge scale image/data processing.

Hadoop is having its own storage system named it as HDFS can store large files across multiple machines. HDFS is a Hadoop distributed File System (HDFS) which uses Master-Slave architecture, as shown in Figure 1. This architecture mainly contains Namenode and Datanode. Master is identified as Namenode contains only metadata of the file system and Slave is identified as Datanode contains Application data is stored in the form of 64MB blocks [5, 18, 19] on multiple Datanodes. Map-Reduce is a parallel programming runs on the upper level of the HDFS consists of one Job Tracker to which client applications submit Map Reduce jobs. The Job Tracker will assign the task to Task Tracker nodes whichever is available in the cluster, maintaining to place the work as near to the data as possible.

Map Reduce function having two modules first one is a Map function, and second is a Reduce function. The input given to the Map function is a key/value pair represented as (k1,v1) and outputs a pair contains intermediate values represented as (k2, v2) The Reduce function accepts all the pairs of values associated with the similar key is represented also as key/value pairs. **Fig. 2** below shows the working mechanism of Map-Reduce.

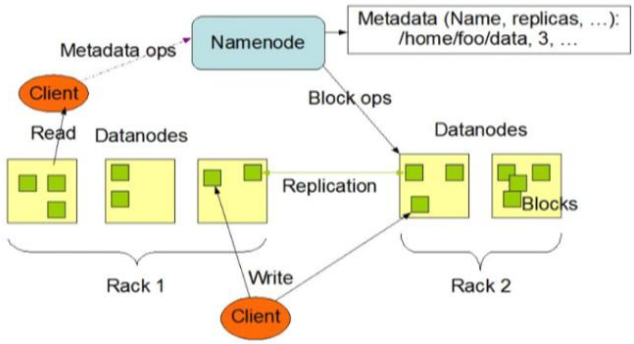


Fig.1: HDFS Architecture.

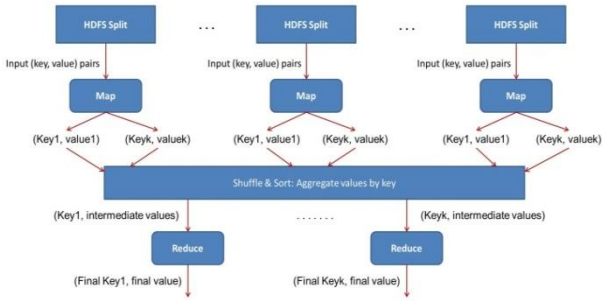


Fig.2: Process of Map Reduce model.

IV. METHODOLOGY

Processing of Higher Resolution Remote Sensing images (HRRS) is a giant task. Many tools are available for processing images in image processing, but the drawback is which are optimized for a single machine and are sequential in nature. In the same manner, when we process high-resolution remote sensing images, it will take a long time. Hence, we need a distributed framework that can work parallel in a most efficient manner because the input data size is much larger than the computation involved in the processing of high-resolution remote sensing images. Based on these facts, to process these images efficiently, the paper proposed a distributed framework for higher resolution remote sensing images using Hadoop Distributed File System (HDFS). **Fig. 3** above shows the distributed framework for the process of HRRS images. HRRS images are split into two forms dimension based splitting and memory-based splitting. After splitting the image, upload the file into HDFS (using a simple put command: hdfsdfs –put <filename>), and process it by Map-Reduce. The process of Map Reduces, as shown in **Fig. 2,** contains key-value pairs as input and output, which are represented in the sequence file format. Hadoop framework saves the images delivered by satellite in HDFS, and the data will be distributed amongst several data nodes.

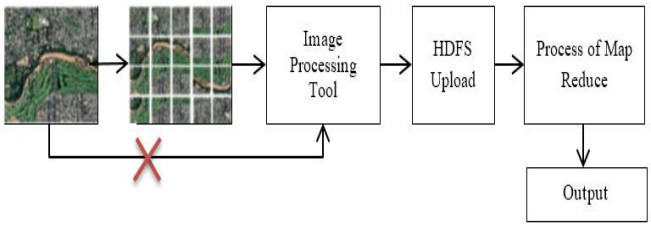


Fig.3: A distributed framework of HRRS Images.

The proposed methods consist of two phases. Phase-1 discuss Pre-processing and Phase-2 discuss the process of Map-reduce function.

Phase-1-Preprocessing: At first, store the HRRS images into HDFS, but these images are split into smaller files which are discussed in Algorithm-1. This splitting process is important for handling large files which are converted into a sequence file then merge all the files which are of size at least 64MB. Every set of images (split) are converted to serialization format. The generated SequenceFiles forms the input to the next Map-Reduce job. MapReduce framework will process all of these files in parallel.

*/*Algorithm-1: Image split*/*
Class: SerializeImageRecordReader
Function: Next
Next(Text key, BytesWritable value)
Call-1: Deserialize_FileSplit<Fi. blockserial>;
For every call
set_key: Filename of serialized image in InputSplit<Fi.blockserial>;
set_value: Data <img_i.serial> of serialized image in InputSplit <Fi. blockserial>;

Phase-2-MapReduce function: Map-Reduce function input for HDFS. The process of MapReduce is shown in Figure 2. Map task will process the block level of serialized files, HFx.bser. These files will be assigned to a Map task that will process the InputSplit. Outputs of the task will be processed by the Reducer task. If required, the next level of the algorithm will be implemented in the Reducer, and the last output is saved to the HBase. The following Algorithms 2 and 3 discusses Mapper class in Map function Reducer class in Reduce function. After the processing phase, we shall store all the files to HBase. The HBase table will assist in the way of accessing image output after processing.

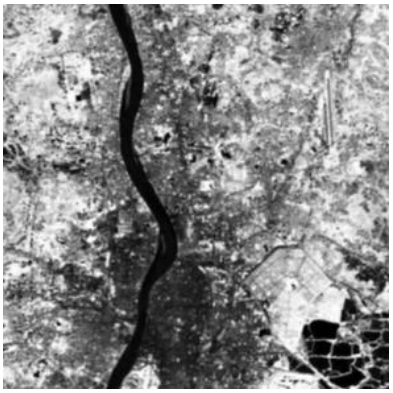
*/*Algorithm-2: Mapper class*/*
Class: Mapper
Function: Map
Map(Text key (filename), BytesWritable value (serialized image data<img_i.serial>), OutputCollector)
Initialize_Javacv;
Initilize_Opencv;
deserialize_value (img_i.serial);
For each Image Split<Tr> in img_i.serial
ImageProcessing(Tr);
Serialized_each_map_Output (OT1 .....OTr) -> OFIi.serial;
OutputCollector:
set_key: Filename (same as key input to Map);
set_Value: serialized map output <OFIi.ser>;

```
/*Algorithm-3: Reducer class*/
Class: Reducer
Function: Reduce
Reduce(Text key (filename), BytesWritablevalue(serialized Map output<OIFi.ser>), OutputCollecto
deserialize_value(OIFi.ser);
Initialize_Javacv;
Initilize_Opencv;
For each processed Image Part<OTr> in OIFi.ser
ImageProcessing (Tr);
Merged_Each_Output(FT1 .....FTr) -> OIr;
Store_Reduce_Output_To_HBase(OIr);
OutputCollector:
set_key: filename (Individual Image filename);
set_value: reduce output <OIr>;
```

For Mapper and Reducer classes JavaCV tool is used for OpenCV functions like Java wrapper. This tool helps for highly efficient image processing algorithms like high-resolution remote sensing images. And another tool is used for the same, i.e., JNI (Java Native Interface). JNI is an interface between OpenCV programs. OpenCV programs are modeled in C or C++ and Java. OpenCV is open source software which provides many libraries which is useful for image processing algorithms.

V. RESULTS AND DISCUSSIONS

Here, the experiments are conducted for remote sensing data of high-resolution images. This data contains large in size either giga bytes or terabytes and also different dimensions and varying sizes. To test and verify the proposed algorithm, different datasets from IRS-1A, and SPOT remote sensing images, including medium and high-resolution images, are used as one set. The IRS-1A image is obtained from the Indian Remote Sensing Satellite [22]. The configuration of IRS-1A image located at Calcutta shown in Fig. 4(a) and SPOT image is shown in Fig. 4(b) as tabulated in Table 2. The IRS-1A image is taken from Linear Imaging Self-Scanner using 4 bands and SPOT image using 3 bands. When we observe both the images, IRS-1A used 4 bands, but they are poorly illuminated when compared to SPOT image of 3 bands. Both the images cover an area into six major classes like pure water, turbid water, concrete area, habitation, vegetation, and open spaces. In this study, we consider another satellite image of High-Resolution image located at Madurai state of Tamilnadu in India is shown in Fig. 5.



(a)

(b)

Fig. 4: Original (a) IRS-1A (band 4) and (b) SPOT (band 3) image.

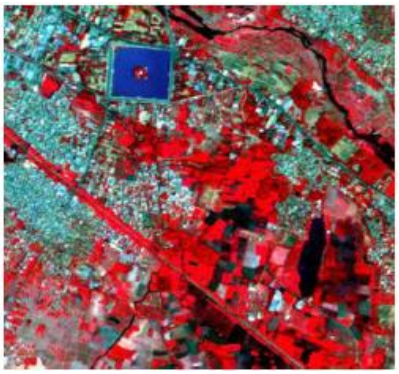


Figure 5: Madurai city in Tamilnadu.

Table 2: Satellite Image dataset located at Calcutta.

Satellite Type	Wavelength	Spatial Resolution	Range
IRS-1A image Figure 4(A)	0.45–0.86 μm	36.25×36.25m	Four bands are Red, Green, Blue and near-infra red
SPOT image Figure 4(B)	0.50–0.89 μm	20×20 m	Three bands (green, red, and near-infra red

IRS P6/LISS IV Resolution Figure 5	0.52~0.59 µm	101 m	Band 2 (green) Band 3 (red): 0.62~0.68 µm Band 4 (near-Infrared): 0.77~0.86 µm

The following configuration is considered for evaluating the performance of the proposed method using HRRS images as tabulated in Table 3.

Table 3: Configuration of the proposed System.

Stage-1	Hadoop-MATLAB integrated cluster	the cluster consists of 30 systems with Master-Slave configuration
Each Machine configuration	single quad-core Intel Core i7 8th generation, processor- 3.6 GHz, 8 GB DRAM memory, and two 1 TB SATA2 7200RPM hard disks	
Operating System	Ubuntu Linux (14.04)-64	
Software: Hadoop Version 2.7.2 is installed for MapReduce platform	Node-1 is configured with job tracker and name node; other nodes are configured as task trackers and data nodes.	

Normally, an image processing software can load the image of size/dimension 2,00,000×2,00,000, but it takes more time to process the image of this dimension directly. So, that's the reason why we are processing the image in the form of small chunks with few dimension, for example, 1000×1000. In another case, if the dimensions of the image are large, for example, 4 GB, the software may not read the entire image to process. So, in such case, we will split the image into small pieces according to the available size of memory, i.e., 256 MB per piece size. In each case, it splits the image on the basis of image dimension or image size [20]. We have created an example image whose size is 3.14GB. When we shall start to upload the image more than that size, it will be not be processed by the software and error is displayed as *OUT OF MEMORY*. Consider HRRS image, image splitting based on dimension into equal-sized blocks of different sizes 1024×1024 pixels and 2048×2048 pixels. Generally, HRRS are large in size and occupies more memory. So processing of these images takes much time. The following Table 4 shows that results of 1024 and 2048 blocks processing using Map-Reduce method. To process 1024 blocks in Phase-1 and Phase-2, it takes 7 minutes 3 seconds and 1 minute 7 seconds respectively. But it is less time compared to standalone mode. For 2048 sized blocks, it takes 7 minutes 3 seconds and 1 minute 7 seconds respectively for 700 images.

Table 4: Time Taken for Processing Images of Block-based and Size based Splitting

	Image Dimension	Run time	Time Taken for Phase-1	Time Taken for Phase-2
Block-based Splitting	1024×1024 blocks	1st time processing	25 Min. 13 Sec.	2 Min. 25 Sec.
		2nd time processing		1 Min. 59 Sec.
		3rd time processing		1 Min. 32 Sec.
		4th time processing		1 Min. 23 Sec
		5th time processing		1 Min. 20 Sec
	Average running time for processing 1024 images =			**1 Min 33 Sec**
	2048×2018 blocks	1st time processing	7 Min. 3 Sec.	1 Min. 15 Sec.
		2nd time processing		1 Min. 9 Sec.
		3rd time processing		1 Min. 6 Sec.
		4th time processing		1 Min. 5 Sec.
		5th time processing		1 Min. 3 Sec.
	Average running time for processing 700 images =			**1 Min 7 Sec.**
Size/Memory	**65 MB**	1st time processing	2 Min. 59 Sec.	2 Min. 25 Sec.
		2nd time processing		1 Min. 59 Sec.
		3rd time processing		1 Min. 32 Sec.
		4th time processing		1 Min. 23 Sec.
		5th time processing		1 Min. 20 Sec.
	Average running time for processing 65 MB size of images =			**1 Min 36 Sec**

VI. CONCLUSION

This paper proposed a distributed framework for processing High-Resolution Remote Sensing Images (HRRS) of IRS-1A and SPOT located in Calcutta. For processing these HRRS images, two phases, namely block-based and size based splitting methods, are implemented. The time taken for execution of the proposed distributed framework was considerably less time when compared to the standalone method. But the proposed method found two problems. The first problem is image splitting based on blocks when we process the image one after another only possible and the second problem is the size of the image will run up to 4 GB size, if size may increase automatically the message will come out of memory. Currently, this framework is applied only on a specific type of images.

Finally, this paper developed a distributed framework using Hadoop and MatLab integrated approach for HRRS images. The proposed method is best fit for complex image processing problems in the coming years due to its consistency.

REFERENCES

1. JiajiWu, LongDeng, AnandPaul, 3D terrain real-time rendering method based on CUDA-Open-GL interoperability, IETE Technical Review ahead-of-print (2015) 1–8.
2. Satellite imaging corporation, http://www/satimagingcorp.com/satellite-sensors
3. N.M.U.Rathore, AnandPaul, AwaisAhmad, BoweiChen, B.Huang, WenJi, Real-time big data analytical architecture for remote sensing application, IEEEJ.Sel.Top.Appl.EarthObs. Remote Sens. 8(7) (2015), 1-12.
4. Kothuri R K V, Ravada S, Abugov D. Quadtree, and R-tree indexes in oracle spatial: a comparison using GIS data[C]//ACM SIGMOD International Conference on Management of Data. ACM, 2002:546-557.
5. http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.
6. http://static.googleusercontent.com/media/research.google.com/es/us/archive/mapreduce-osdi04.pdf.
7. Zheng K, Fu Y. Research on Vector Spatial Data Storage Schema Based on Hadoop Platform[J]. International Journal of Database Theory & Application, 2013, 6(5):85-94.
8. http://wiki.apache.org/hadoop/Hbase
9. Rajak R, Raveendran D, Bh M C, et al. High-Resolution Satellite Image Processing Using Hadoop Framework[C]//IEEE International Conference on Cloud Computing in Emerging Markets. IEEE, 2015:16-21.
10. L. Kennedy, M. Slaney, and K. Weinberger, Reliable Tags using Image Similarity: Mining Specificity and Expertise from Large-Scale Multimedia Databases, in Proc. 1st workshop on Web-Scale Multimedia Corpus, Beijing, October 2009, pp. 17-24.
11. L. L. Shi, B. Wu, B. Wang, and X. G. Yang, Map/Reduce in CBIR Applications, in Proc. International Conference on Computer Science and Network Technology (ICCSNT), Harbin, December 2011, pp. 2465-2468.
12. Mohamed H. Almeer, "Cloud Hadoop Map Reduce For Remote Sensing Image Analysis" Journal of Emerging Trends in Computing and Information Sciences, Vol. 3, No. 4, April 2012
13. H Kocakulak and T T Tamizel, "A Hadoop solution for ballistic image analysis and recognition" International Conference on High-Performance Computing and Simulation (HPCS), Istanbul, pp. 836-842, 2011.
14. Y. Ma, H. Wu, L. Wang, B. Huang, R. Ranjan, A. Zomaya, and W. Jie, Remote Sensing Big Data computing: Challenges and opportunities, Future Generation Computer Systems, Volume 51, October 2015, Pages 47–60.
15. B. Li H Zhao, Z H Lu, "Parallel ISODATA clustering of remote sensing images based on MapReduce," International Conference Cyber-enabled distributed computing and Knowledge Discovery (CyberC), Huangshan, pp. 380-383, 2010.
16. Z. Lv, Y. Hu, H. Zhong, J. Wu, B Li, and Z Zhao, 2010 "Parallel K means clustering of remote sensing images based on MapReduce," International Conference on Web Information Systems and Mining, pp. 162-170. 2010.
17. P. Jakovits, and S. N. Srirama, Large Scale Image Processing Using MapReduce, thesis, Tartu University, 2013.
18. Roshan Rajak, Deepu Raveendran, Maruthi Chandrasekhar, and Shanti Swarup Medasani, High-Resolution Satellite Image Processing using Hadoop framework, 2015 IEEE International Conference on Cloud Computing in Emerging Markets, 978-1-4673-8566-4/16.
19. Süleyman Eken, Eray Aydin, Ahmet Sayar, DIFET: Distributed Feature Extraction Tool For High Spatial Resolution Remote Sensing Images, ISPRS Journal of Photogrammetry and Remote Sensing • November 2017, pp: 209-2013.
20. U.S.N.Raju, Suresh Kumar, Vedpratap Mehta, Rishabh Sharma, Sushanth Kuli, Cluster-Based Block Processing for Gigantic Images: Dimension and Size, 2017 Fourth International Conference on Image Information Processing (ICIIP), pp: 552-556.
21. Jiao Shi, Jiaji Wu, Anand Paul, Licheng Jiao, Maoguo Gong, Change detection in synthetic aperture radar image based on fuzzy active contour models and genetic algorithms, Math.Prob.Eng.2014(2014)15.
22. Rajesh S. Arivazhagan Selvaraj, Pratheep Moses, R. Abisekaraj, Land Cover/Land use Mapping Using Different Wavelet Packet Transforms for LISS IV Madurai Imagery, Journal of the Indian Society of Remote Sensing 40(2):313-324, January 2012.
23. Venkateswar Lal, P., Nitta, Gnaneswara Rao. & Prasad, A.," Ensemble of texture and shape descriptors using support vector machine classification for face recognition" Journal of Ambient Intelligence and Human Computing, (2019). https://doi.org/10.1007/s12652-019-01192-7 .
24. Nitta, Gnaneswara Rao., Sravani, T., Nitta, S. et al.," Dominant gray level based k-means algorithm for MRI Images," Journal of Health and Technology, (2019). https://doi.org/10.1007/s12553-018-00293-1 .
25. Gnaneswara Rao Nitta, Yogeshwara Rao B Sravani T Ramakrishiah N," LASSO based Feature Selection and Naïve Bayes classifier for crime prediction and its type," Service Oriented Computing and Applications, Springer, ISSN: 1863-2394, Impact factor:1.40, H-Index:15, (2019).
26. A. Sindhura, S. Deva Kumar, V Ramakrishna Sajja, N Gnaneswara Rao, "Identifying Exudates from Diabetic Retinopathy Images," 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), ISBN No.978-1-4673-9544-1, Organized by Syed Ammal Engineering College, Ramanathapuram, Tamilnadu, India.
27. Tejaswi Potluri, Gnaneswararao Nitta, "content-based video retrieval using the dominant color of the truncated blocks of the frame," Journal of Theoretical and Applied Information Technology," Vol.85. No.2 © 2005 - 2016 JATIT & LLS, ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195.
28. N.Gnaneswara Rao, T.Sravani, and V.VijayaKumar, "OCRM Optimal Cost Region Matching Similarity Measure for Region-Based Image Retrieval," International Journal of multimedia and ubiquitous Engineering, Vol.9, No.4 (2014), pp.327-342.

AUTHORS PROFILE

T. Nagaraju received the Master's degree in KIET Engineering College and is currently pursuing the Ph.D. degree at Acharya Nagarjuna University, Guntur. His research interests include Big Data analytics, Image Processing, and network security.

Dr. Chittineni Suneetha is working as Associate Professor in the Dept. of Computer Applications. She has 19 years of teaching experience. Her areas of interest are Soft Computing, Artificial Intelligence, Image Processing, and Big Data. Currently her publications are 35 international journals and 18 international Conferences.