Content Based Image Retrieval with Hadoop

Heba Gaber(✉), Mohammed Marey, Safaa E. Amin, and Mohamed F. Tolba

Computer and Information Sciences, Ain Shams University, Cairo, Egypt
eng.heba.gaber@gmail.com, mohammedmarey@hotmail.com, safaa_amin007@htomail.com, fahmytolba@gmail.com

Abstract. Hadoop has become a widely used open source framework for large scale data processing. MapReduce is the core component of Hadoop. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. It allows processing of extremely large video files or image files on data nodes. This can be used for implementing Content Based Image Retrieval (CBIR) algorithms on Hadoop to compare and match query images to the previously stored terabytes of an image descriptors databases. This work presents the implementation for one of the well-known CBIR algorithms called Scale Invariant Feature Transformation (SIFT) for image features extraction and matching using Hadoop platform. It gives focus on utilizing the parallelization capabilities of Hadoop MapReduce to enhance the CBIR performance and decrease data input\output operations through leveraging Partitioners and Combiners. Additionally, image processing and computer vision tools such as Hadoop Image Processing (HIPI) and Open Computer Vision (OpenCV) are integration is shown.

Keywords: Hadoop · HIPI · OpenCV · Big data · Crowd sourcing · MapReduce · Combiners · Partitioners

1 Introduction

CBIR Systems are used for searching and retrieving query images from large databases based on the image content, which is derived from images themselves by using computer vision techniques. SIFT is one of the most important CBIR techniques that depends on extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. For large scale images or large numbers of images stored in the database, SIFT feature extraction is a computationally intensive problem, as it takes a long time to extract and match them to the extracted SIFT features. Therefore, there exists a need for an efficient platform to handle image descriptor features extraction and processing that may reach Terabytes of data in size.

In this work Hadoop is used as a reliable, scalable, distributed computing platform. It is an open source project from the Apache Software Foundation [14, 15] and its core consists of MapReduce programming model implementation. Hadoop enables the execution of applications on thousands of nodes and petabytes of unstructured or semi-structured data which have been impossible to process efficiently (cost and time) so far.

A.E. Hassanien et al. (eds.), *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016*, Advances in Intelligent Systems and Computing 533, DOI 10.1007/978-3-319-48308-5_25

This paper presents the implementation of the SIFT descriptor extraction and matching using Hadoop MapReduce, integrated with HIPI and OpenCV libraries for SIFT features extraction and matching. Moreover, we show how to enhance the matching performance by leveraging parallelization mechanisms for MapReduce Partitioners and Combiners.

The paper is organized as follows: Sect. 2 discusses the related work, Sect. 3 presents CBIR with SIFT algorithm, Sect. 4 presents Hadoop MapReduce, Combiners and Partitioners. Section 5 discusses the integration of HIPI and OpenCV computer vision libraries utilized in the proposed system. Section 6 explains implementing CBIR with SIFT for image processing on Hadoop and how to enhance its performance, Sect. 7 concludes the presented work and shows our future research direction.

2 Related Work

Recently, using Hadoop for image processing has been receiving attention, there exists some recent implementations for many domains like biomedical, social media, surveillance, satellite and geographical images processing applications summarized in Table 1.

In [1] a machine learning tool "HaarFilter" is implemented detecting human faces by using the Haar Cascading technique. It was implemented using HIPI and OpenCV on Hadoop platform. Compressing social media images where HIPI is used as image processing tool as shown in [2]. Providing image processing as a service published to the public and implemented on Hadoop platform using integration for OpenCV library is presented in [3]. In [4] feature extraction using Hadoop and SIFT is shown.

Image processing with Hadoop for health care in India using HIPI that allows is presented in [5]. Using HIPI and Avro for processing surveillance images is shown in [6]. In [7], analyzing terabytes of microscopic medical images on Hadoop using Tera-soft library is demonstrated. Processing satellite and geospatial huge images databases on Hadoop is shown in [8, 9].

Table 1. Image processing with Hadoop platform

Ref	Tools	Application
[1]	OpenCV, HIPI	Machine learning for face detection using Hadoop
[2]	HIPI	Compress social media images
[3]	OpenCV	Provide image processing tools as a service
[4]	Custom Development	Feature extraction (target tracking, traffic management and accident discovery) using Hadoop and SIFT
[5]	HIPI	Image processing with Hadoop for application of heath care applications in India
[6]	HIPI, Avro	HIPI and Avro for processing surveillance images
[7]	Tera-soft	Analyzing terabytes of microscopic medical images
[8]	Custom	Satellite image processing
[9]	Custom	Processing 100 M Geographical images on Hadoop Geographical files processing

3 CBIR with SIFT

CBIR with SIFT can be implemented as illustrated in Fig. 1 using four main components (1) Image collection, (2) Image features extraction, which mainly affects the quality of searching, (3) Features Indexing and (4) Searching using Query images, which mainly affects the efficiency.

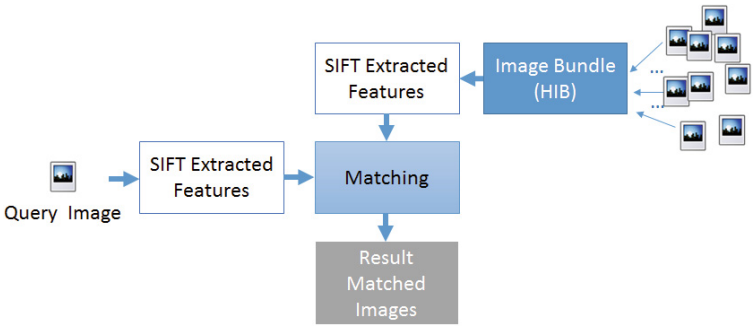

Fig. 1. CBIR with SIFT Architecture

SIFT algorithm can be implemented effectively for content based image retrieval based on Region of Interest (ROI). The SIFT feature is invariant to rotation, image scaling and transformation and partially invariant to illumination changes and affine transformation. The main advantage of using SIFT is that it describes the same features on different spatial scales.

SIFT is based on convolving an image with a Gaussian kernel for several values of the variance σ and then taking the difference between adjacent convolved images. If $D(x, y, \sigma)$ denotes such differences, then the method is based on finding the extrema of this function with respect to all the three variables. Such maxima are candidates for key points that, after additional analysis, are used to characterize objects in an image. SIFT calculate result descriptor is shown in Fig. 2 [10]. DoG $D(x, y, \sigma)$ provides a good approximation for the Laplacian-of-Gaussian. It can efficiently be computed by subtracting adjacent scale levels of a Gaussian pyramid.

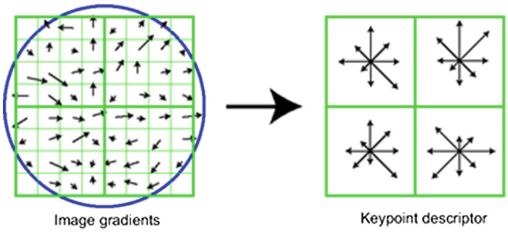

Fig. 2. SIFT Descriptor Generation [10]

SIFT feature extraction is composed of four important phases:

- Phase 1- Scale-space Extrema Detection: potential interest points are identified by searching the overall scales and image locations in the image matrix. The scale space of an image is defined as a function $L(x, y, k\sigma)$ and produced from the convolution of a variable scale Gaussian $\mathrm{G}(x, y, k\sigma)$, with an input image, $I(x, y)$

$$L(x, y, k\sigma) = \mathrm{G}(x, y, k\sigma) * I(x, y) \tag{1}$$

$$D(x, y, \sigma) = (L(x, y, k\sigma) - L(x, y, \sigma)) * I(x, y) \tag{2}$$

- Phase 2 Keypoint localization: A detailed model is created to determine location and scale of all interest points detected in phase1. After that, key points are selected based on their stability and their resistance to distortion.
- Phase 3- Orientation assignment: For each key point identified in phase 2, the direction of gradient is computed around it. One or more orientations are assigned to each key point based on local image gradient directions.
- Phase 4: Keypoint descriptor: In this phase, the local image gradients are measured in the region around each key point. This gradient is then transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

One of the most featured implementations for CBIR is vision based Navigation, based on a Visual Information System (NAVVIS) project for indoor navigation for the Technical University of Munich (TumIndoor). The navigation application uses images that are captured by a smart phone as visual fingerprints of the environment. The captured images are matched to the previously recorded geotagged reference database with CBIR techniques. TumIndoor introduces an extensive benchmark dataset for visual indoor localization, available to the research community for downloading, thus any experimental results can be compared with the published results [10, 11]. Some other feature extraction and matching techniques exist such as; Speed-Up Robust Features (SURF) [12] and Binary Robust Independent Elementary Features (BRIEF) [13]. SIFT's accuracy holds up against more modern algorithms yet it's computationally expensive.

4 Hadoop, Map-Reduce, Combiners and Partitioners

The two main components of Hadoop are: Hadoop Distributed File System (HDFS) and MapReduce. MapReduce paradigm is composed of three phases. (1) The Mapper: Each Map task operates on a single HDFS block and runs on the node where the block is stored. (2) Shuffle and Sort: Sorts and consolidates intermediate data from all mappers, after the Map tasks are completed and before the starting of the Reduce tasks. (3) The Reducer: Operates on shuffled/sorted intermediate data (Mapping output) to produce the final output.

The Combiner phase in MapReduce can enhance the cluster performance by reducing results on a single Mapper's output before sending the data to the reducer.

A Combiner, also known as a semi-reducer, which is an optional class that operates by (1) accepting the inputs from the Map class as key value pairs, (2) summarizing the map output records with the same key, (3) passing its output as new key-value pairs to the original Reducer class.

The Cluster performance can also be enhanced through partitioning (indexing) the keys using custom or default Partitioner, where each partition is passed to a single reducer. A Partitioner partitions the key value pairs of intermediate Map outputs. It partitions the data using a user defined condition, which works like a hash function. The total number of partitions is the same as the number of Reducer tasks for the job.

In Fig. 3 using Hadoop MapReduce Paradigm by integrating HIPI and OpenCV libraries for extracting SIFT features is presented, the first step is collecting image data-base into a HIB bundle, afterwards OpenCV is used for extracting image data, pre-processing (grayscale conversion) and feature extraction afterwards. In this work we show how processing data in combiner and Partitioner phase and changing the number of processing units can effectively enhance the overall performance and reduce the input/output operations of the system.

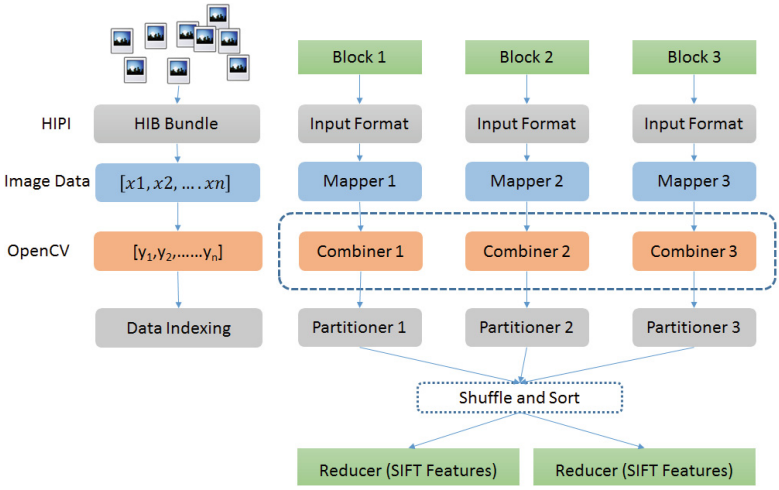

Fig. 3. Map-Reduce Paradigm

5 Integrating Computer Vision Libraries

For image processing on a Hadoop platform, HIPI was introduced. HIPI [15] is an image processing library designed to be used with the Apache Hadoop Map-Reduce parallel programming framework. HIPI facilitates efficient and high throughput image processing with Map-Reduce style parallel programs typically executed on a cluster. It provides a solution for how to store a large collection of images on the HDFS and make them available for an efficient distributed processing. HIPI is developed and maintained by a growing number of developers around the world.

The primary input object to a HIPI program is a HIPI Image Bundle (HIB) which is a collection of images represented as a single file on the HDFS [16], the first processing stage of a HIPI program is a culling step that allows filtering the images in a HIB, based on a variety of user defined conditions like spatial resolution or criteria related to the image metadata.

Further processing can be done using OpenCV [17] for feature extraction. We selected the widely used OpenCV library as the base image processing library integrated with the proposed platform. OpenCV is an open source library written in C++ and it also has Java and Python interfaces supporting Windows, Linux, Mac OS, iOS and Android. OpenCV is optimized and parallelized for multi-cores and accelerators using the Open Computer Library (OpenCL). OpenCV was installed on the Hadoop cluster to enable image processing capability with Map-Reduce parallel programming model.

In the presented work integrating HIPI and OpenCV was introduced, the images folder will be uploaded to the HDFS as a HIB bundle using HIPI, afterwards in the feature extraction job, OpenCV library is used to get SIFT features from the HIB. The result of the feature extraction job will be saved on Hadoop. Another job for matching query image to the descriptors database was implemented to find the matched image.

6 SIFT Based CBIR Implementation on Hadoop

The proposed approach mainly depends on CBIR techniques using SIFT to match query images with image databases. In this implementation the Technical University of Munich (TumIndoor) dataset used for the Indoor navigation application is selected to test the proposed approach. The database used in our implementation is constructed of 9,437 images for one floor, total size of the images is 1,395,864,371 bytes.

The first step of the implementation was uploading images on HDFS and bundling images utilizing HIPI. The size of the constructed HIB bundle using HIBI is 1,503,238,553 bytes. The total size of the generated descriptors is 8,877,244,416 bytes. Afterwards a database of image SIFT descriptors is constructed with Map-Reduce job using OpenCV library. Moreover data input/output operations reduce through using Combiner intermediate layer, data was processed on the Combiner phase which enhanced the overall systems performance.

For matching query images with the descriptor database, another matching job is implemented. The matching job extracts the descriptors of the query image and matches them to the descriptors database by using OpenCV library. The matching job's runtime performance was tested with different numbers of reducers and was enhanced through grouping the descriptors based on the descriptors' number of rows in-stead of using the image name as the mapper output key.

The implementation performed on Cloudera virtual machine [18] where Java Eclipse and Hadoop were already installed and HIPI and OpenCV is configured. The current machine consists of 4 processors and 10 GB RAM. By adding more memory, processors and additional physical nodes the run-time performance of the system will increase and the processing time is expected to decrease. The algorithm is tested with different parameters on the same environment by changing the number of reducers and adding Combiners and Partitioners on different phases of computation. We noticed that grouping the data under 100 reducers distributed based on the number of the rows in each descriptor

Table 2. HDFS and File execution KPIs

Environment Configuration/Counter	FILE: Number of (GB) read	FILE: Number of (GB) Written	HDFS: Number of bytes read	HDFS: Number of read operations
1000 reducer	16.56	24.46	28.4	12,650
100 reducers	16.14	24.36	28.4	9,950
50 reducer	16.12	24.35	28.4	9,800
100 reducer and classifier num-rows	16.14	24.36	12.8	2,631
Partitioner and 100 reducer with classifier num-rows %100	16.14	24.36	12.8	2,631
Partitioner, combiner and 100 reducers classifier num-rows %100	0.0036	0.018	19.17	5,618
Partitioner, combiner and 100 reducers classified by num-rows	0.0036	0.018	21.05	6,499

matrix enhanced the performance more than using very large number of reducers (1000) or small number of reducers (50). This can't be taken for granted as the optimal number of reducers and the partitioning parameter shall depend on the input data size and the variation of the size of the generated descriptors. We also show the impact of adding Combiners and the Partitioners after grouping the data under a 100 reducer.

Hadoop offers Job Tracker, an UI tool to determine the status and statistics of all jobs. Using the job tracker UI, we can view the Counters that have been created and investigate the execution results. Table 2 shows the results of the comparison between different execution parameters in terms of mapping time, reduced shuffling processing time, and numbers of read and write operations and overall performance of processing the same dataset. Examples for those Counters are:

1. *File-number of bytes read:* is the number of bytes read by local file systems and it denotes the total bytes read by reducers. They also occur during the shuffle phase when the reducers spill intermediate results to their local disks while sorting.
2. *File-number of bytes written:* consists of two parts. The first part comes from mappers. The second part comes from reducers. In the shuffle phase, all the reducers will fetch intermediate data from mappers and merge and spill to reducer-side disks. All the bytes that reducers write to disk will also be included in this counter.
3. *HDFS-number of bytes read:* Denotes the bytes read by mappers from HDFS when the job starts. This data includes not only the content of source file but also metadata about splits.
4. *HDFS-number of read operations:* Denotes the bytes written to HDFS. It's the number of bytes of the final output. Since HDFS and local file systems are different file systems, so the data from the two file systems will never overlap. From this

implementation, the least file system access was achieved through using Partitioners and Combiners so that data is accessed after data grouping in the Map Phase.

The other performance KPIs are related to the time and the memory consumed in each phase in the Map-Reduce. Total time spent by all maps (5), All reducers (6), All map tasks (7), All reduce tasks (8) in milliseconds, total megabyte seconds taken by all map tasks (9), All reduce tasks (10). It was noticed the least mapping time was achieved through partitioning the data with descriptor metadata element ex: "Number of rows in the SIFT descriptor matrix" and without using Partitioners or Combiner layers, this can be explained as the data won't be re-copied after the mapping phase. On the opposite side the least reduce time was achieved through adding Combiners and Partitioners and getting reduced input/output through the intermediate layer before Shuffle and Sort process.

Comparing the two approaches the second approach is more efficient since mapping time can be enhanced through adding more slots to the system and distribute processing on them, yet the reduce phase is considered a critical phase as it will be responsible to process all the Mappers output to get the final results and sometimes considered a bottleneck phase.

Also, we noticed enhanced performance counters by using Partitioners and Combiners for example (11) GC (Global Cash) time elapsed, (12) Processing time (CPU time spent), (13) Data snapshot physical and (14) Virtual memory and the (15) Total heap usage.

7 Conclusion and Future Steps

In this work we implemented CBIR based on SIFT for features extraction and matching of query images to huge image descriptors on Hadoop. SIFT is very intensive and requires very high processing and storage power. In this work Hadoop MapReduce is used as a Big Data platform for processing and storage of the image descriptors data. HIPI is integrated in the system and used to upload and filter data to HDFS as HIB bundle. This work also shows integrating HIPI and OpenCV computer vision libraries with Hadoop to accelerate features extraction and matching. The Feature extraction job was implemented to generate image descriptors database based on SIFT features. Another job was implemented to match query images to the database. We have shown how to use Map-Reduce Combiners and Partitioners for enhancing performance and reduce the input/output operations. The proposed approach seeks to maximize the full potential of cloud computing as well as ubiquitous sensing, a viable way to achieve this was to have a combined framework with a cloud at the center.

We have demonstrated the practicality of our approach by extracting SIFT features from TumIndoor images, confirming that the Map-Reduce based approach can accelerate the feature extraction process effectively through a Map-Reduce paradigm for a database of about 8 GB in size. Our next step is to enhance the matching with the SIFT descriptors through categorizing the data as visual words and involving machine learning techniques to classify the bag of visual words.

References

1. Nausheen, K.M., Ram, M.S. Haarfilter: a machine learning tool for image processing in Hadoop. Int. J. Technol. Res. Eng. 3 (2015)
2. Barapatre, M.H., Nirgun, M.V., Jagtap, M.H., Ginde, M.S.: Image processing using mapreduce with performance analysis. Int. J. Emerg. Technol. Innovative Eng. I(4) (2015)
3. Yan, Y., Huang, L.: Large scale image processing research cloud. In: Cloud Computing, pp. 88–93 (2014)
4. Cheng, E.: Efficient feature extraction from a wide area motion imagery by MapReduce in Hadoop. In: SPIE Defense + Security. International Society for Optics and Photonics (2014)
5. Augustine, D.P.: Leveraging big data analytics and hadoop in developing India's healthcare services. Int. J. Comput. Appl. **89**(16), 44–50 (2014)
6. Gawde, A.U., Shah, M., Ukaye, I., Nanavati, M.: Object detection in hadoop using HIPI. Int. J. Adv. Res. Eng. Technol. (2013)
7. Bajcsy, P.: Terabyte-sized image computations on Hadoop cluster platforms. In: IEEE International Conference on Big Data, pp. 729–737. IEEE (2013)
8. Han, W., Kang, Y., Chen, Y., Zhang, X.: A MapReduce approach for SIFT feature extraction. In: International Conference on Cloud Computing and Big Data, pp. 465–469 (2013)
9. Moise, D., Shestakov, D., Thor, G., Amsaleg, L.: Indexing and searching 100 M images with Map-Reduce. In: ACM International Conference on Multimedia Retrieval, pp. 17–24 (2013)
10. Huitl, R., Schroth, G., Hilsenbeck, S., Schweiger, F., Steinbach, E.: TUMindoor: An extensive image and point cloud dataset for visual indoor localization and mapping. In: 19th IEEE International Conference on Image Processing (ICIP) Orlando, FL, pp. 1773–1776. IEEE (2012)
11. Schroth, G.: Mobile Visual Location Recognition. Ph.D. Thesis. Munich: Technische Universität München, July 2013
12. Panchal, P.M., Panchal, S.R., Shah, S.K.: A Comparison of SIFT and SURF. Int. J. Innovative Res. Comput. Commun. Eng. **1**(2), 323–327 (2013). ISSN: 2320–9798
13. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: binary robust independent elementary features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 778–792. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15561-1_56
14. http://hadoop.apache.org
15. White, T.: Hadoop: The Definitive Guide, 2nd edn. O'Reilly Media, Sebastopol (2011)
16. http://hipi.cs.virginia.edu/
17. http://opencv.org/
18. http://www.cloudera.com/