MapReduce for High-Speed Feature Identification for Computer Vision Applications

Shaheena Noor
PhD Scholar at Computer Engineering
Hamdard University,
Karachi, Pakistan.
shanoor@ssuet.edu.pk

Prof. Dr. Vali Uddin
Dean at Faculty of Engg. Sciences & Technology
Hamdard University,
Karachi, Pakistan.
vali.uddin@hamdard.edu.pk

***Abstract*— This paper presents a MapReduce-based implementation of using high-dimensional image streams applied to feature identification to provide fast computer vision applications. We argue that feature identification is a preliminary step of many computer vision tasks like object recognition, activity understanding, content-based retrieval etc. which is computationally expensive and often becomes a bottle-neck in real-time applications. We considered the state-of-the-art feature extraction method - Scale Invariant Feature Transform (SIFT) - and used MapReduce to parallelize the computation. We considered the sandwich making dataset from TUM, comprising of 1.94 m images. Feature detection on a local machine took an estimated of 46.9576 days while using MapReduce we were able to process the same in much reduced time: 1.46 days on a 8-node cluster and 0.586 days using 20 machines on Amazon cloud. Hence, we achieved the same performance with a speedup of 32 and 80 on the two systems using Mapreduce. The major contribution of this paper is that we proposed a method to use MapReduce on cloud to increase computational speed of feature extraction and matching for computer vision applications, without incorporating the additional cost, resource-consumption and expertise incurred due to parallelization.**

Keywords—MapReduce; SIFT; Computer Vision, Cloud computing, AWS

I. INTRODUCTION

As the computing devices proliferate into our everyday lives, and we inch towards the pervasive computing paradigm, we notice a broad range of innovative computing methods and their applications. Technological globalization and advances have led to a wide variety of modalities of collaborative computing i.e. the use of multiple computing agents to solve individual computational problems. The concept of interdisciplinary exchange of information and collaboration has resulted in a mushroom of innovative computing applications. We see computer vision and image processing joining hands with distributed computing; finance, economics and business intelligence meeting data mining and big data analytics; and biology and medicine hand-in-hand with computers. Thus, many issues that seem daunting within a particular domain are easily solved when integrated with others.

As a scientific discipline, computer vision is the branch of computing that extracts meanings out of visual data i.e. images and videos. This data can be coming from single or multiple cameras, covering static or dynamic scenes with changes in viewpoints, illumination and/or scale; and may be noisy with a lot of clutter. Interpreting the scenes on the basis of this data and extracting information like object recognition and tracking, activity recognition etc. can be challenging and time-consuming in the real-world and involves a number of pre-processing steps. Consider Figure 1 that shows the general computer vision and image-processing framework. As can be seen, all major computer vision and image processing applications begin with some form of pre-processing including feature extraction [1, 2]. As the size of data grows, this prerequisite step becomes more and more time consuming, and can become the bottle-neck in many real-time applications. One of the solutions is to move towards parallelization and process multiple images at once. The major motivation behind the parallelization is that many advances in high performance computing are occurring at an ever-increasing pace. Especially with cloud computing emerging as an alternative to supercomputers and taking out the inherent complexity of parallel processing and programming from the equation, has enabled researchers across diverse fields to explore computation intensive solutions, which were otherwise not possible. When deciding to exploit the parallelizing potential in computer vision tasks, one can look into a number of factors like parameters, images or pixels etc.

It turns out that the problem of handling one microscopic image (e.g. 86273 by 81025 pixels 78.12 GB) is totally different from the scenario of having a larger data set of smaller images (e.g. 48469 images @ 6.5 MB/image 308 GB). While, in the first case, the major problem is memory management, in the later it is more about processing efficiency. This calls for a platform where the task at hand can be divided into smaller chunks and assigned to different processing units that can work in parallel and distributed environment.

The paper is organized as follows: We present a review of existing literature on feature extraction and parallelism for computer vision applications in Section II. Next, we give details of SIFT in Section III. Sections IV and V introduce the distributed computing paradigms and MapReduce for

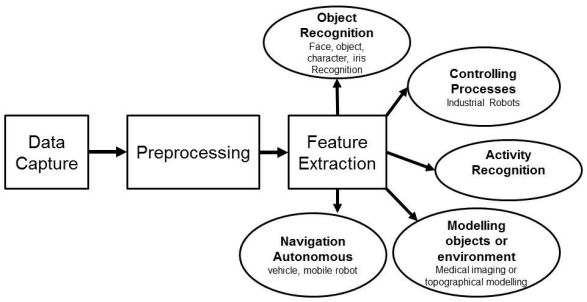

Fig. 1: General Image Processing Framework

computer vision applications. We discuss the experiments and observations in Section VI and present the conclusions in Section VII.

II. LITERATURE REVIEW

Feature Extraction: One of the most important tasks in computer vision and image processing is to extract features from an image [1]. Features efficiently represent the image in the form of unique interest points. Extracting features is useful because it reduces the size of data, and the subsequent computations consume less time. Feature extraction can be based on shape, color, appearance etc. and may be represented as histogram, Bag of Features [2] etc.

Shape detection [1]–[3] represents the information about the geometric shape of an image. On the other hand, color histogram is the representation of color values with the number of pixels at each value. Typically a global histogram is computed for the complete image. The main drawback of this approach is the loss of regional information. As an alternative, recent research uses grid based histogram by dividing the image into sub region. The initial work in point a detector was carried out by Moravec [4], which formed the basis for the detection of salient points around the corners of an object. Some other point detector like Harris [5] and Smallest Univalue Segment Assimilating Nucleus (SUSAN) [5]–[7] for corner detection do not form a descriptor. The process of matching them is purely location based. They are typically used for motion detection, image registration [8], video tracking [9], image mosaicing [10], panorama stitching [11], [12], 3D modeling and object recognition [13], [14]. Xiao and Shah [15] introduced an affine invariant feature namely edge-corner and it was based on Singular Value Decomposition (SVD) of affine matrix that was robust for the matches over two images. Tuytelaars and Van Gool [16] developed an opportunistic system to exploit multiple invariant regions and combine them to form a robust detector. Matas et al. [17] introduced the Maximally Stable Extremal Regions, that handle the significant change in scale, illumination conditions, out-of-plane rotation, occlusion, locally anisotropic scale change and 3D translation of the viewpoint. Kadir et al. [18] developed an interest point detector capable of handling generalized invariance to affine transformation and superior insensitivity to perturbations and intra-class variation performance for images of certain object classes. One of the commonly used and reliable detection mechanisms is SIFT (Scale Invariant Feature Transform), introduced in 1999 by D.G. Lowe [19]–[23] which embeds an image segment descriptor and is widely used in a much complex environment like object recognition and large baseline matching. It is scale and rotation invariant, gives robust results for the changes in illumination and partially invariant to different viewpoints. Over the past many years, many algorithms have been proposed with SIFT as their basis. Some of the renowned ones are Speeded Up Robust Features (SURF) [24], Maximally Stable Extremal Regions (MSER) [17], Gradient Location and Orientation Histogram (GLOH) [25], Local Energy-based Shape Histogram (LESH) [26], Binary Robust Independent Elementary Features (BRIEF) [27], Binary Robust Invariant Scalable Keypoints (BRISK) [28], Fast Retina Keypoint (FREAK) [29], Features from Accelerated Segment Test(FAST) [30] and Oriented Fast and Rotated BRIEF (ORB) [31]. Each of them targets a specific performance aspect like speed, number of features, quality of detection or description, but suffers from one or other limitation. Hence, even after almost two decades, SIFT is still widely used for feature detection. It returns a large number of keypoints (in the order of 100s) and gives reliable matches. However SIFT suffers from slow speed and is, therefore, not suitable for real time applications. For these reasons we used SIFT for our experiments.

Parallelism for Computer Vision Applications: Here we describe the work done on distributed computing for image processing and computer vision. We will not go into the details of parallel computing and programming paradigms, rather only cover their applications to computer vision tasks. With the cost of capturing, uploading and storing of image and video- data going down for the end-user, the data archives are getting much larger than before. To make the most of this data, we need paradigms that make it easier and faster to process this huge amount of data. In recent years, attention has been diverted to using distributed computing for processing and indexing large-scale image datasets [32]–[36]. Previously, researchers would develop, maintain and use their own clusters or grids to solve their computation problems. E.g. Seinstra et al. in [37]–[39] recognized the problem of ever-growing dataset size and proposed a color-based object recognition solution on an in-house grid. As cloud computing became more popular, researchers started using it, so as to transfer the load of setup and programming complexity off from themselves. For example, in [40], Beksi et al. proposed a cloud-based object recognition for robots. Another work by Kehoe et al. [41], where they proposed the thin-client model to move the complex computations to the Google cloud [42] and communicate with the field devices (robot in their) online. They were able to recognize and grasp simple household objects using a Personal Robot (PR2) [43]. In [44], Paul and Park performed multi-view object recognition using their smart phone as a thin client and a computer server as a cloud. They used their smartphone for image acquisition, developed a codebook using the SIFT features and applied Bayes classification for the final recognition. In a similar work, Lorencik et al. [45] used SIFT, SURF and ORB for feature extraction and used the Membership Function ARTMAP (MF ARTMAP) and Gaussian Random Markov Field model for object recognition over cloud.

Classic solutions call for fine-grained control using Message Passing Interface (MPI), however, it becomes overly complicated. More recently researchers have started going for divide-and-conquer approach and MapReduce falls under this paradigm. The idea is to divide the huge data set into smaller chunks and use the distributed cloud computing platforms to process it using regular, low cost and low power computing resources. The MapReduce framework is originally developed by Google [46] and it is a programming model with the supporting implementation for processing extremely large datasets. It takes away the intricacies of parallel processing and programming and the user has to simply specify two functions: the map and the reduce. The Map function processes a key/value pair to generate a set of intermediate key/value pairs, and the Reduce function combines all intermediate values associated with the same intermediate key to give a final output. MapReduce is based on earlier parallelization models e.g. [47] with the additional advantages of providing a fault-tolerant implementation that scales to thousands of processors. In recent years, attention has been diverted to use MapReduce to solve computer vision problems e.g. face tracking [48], landmark classification [49] and visual tagging in photos [50]. In [51], Markonis et al. used Mapreduce for medical image analysis. They developed an in-house cluster and ran MapReduce to address a number of image-analysis scenarios like parameter optimization for Support Vector Machines, image indexing using SIFT and 3D feature texture extraction. However, their major focus was on developing the cluster and parallelizing tasks for increased speed. This is different from our work in several ways. First, our approach is generic and can be applied to a number of indoor and outdoor scenarios. Moreover, we primarily focus on demonstrating that including additional streams from different viewpoints results in increased precision, at the cost of computational time and we handle the time cost by parallelization using MapReduce. Another work by Han et al. [52] presents a MapReduce approach for SIFT extraction. They reformed the original SIFT using MapReduce to make the extraction faster. The benefit of this approach is, however, evident in cases involving high definition images with high resolution and large number of pixels. Our scenario basically deals with large number of low to medium- resolution images coming from multiple streams and thus defines a totally different use case.

There are a number of scenarios where MapReduce is not the best choice, e.g. DeWitt et al. [53] compared MapReduce with parallel database query systems and argued that it is much less sophisticated and efficient. According to [54], MapReduce model imposes strong assumptions on dependence of data and its results are based on these assumptions. Chu et al. in [55] pointed out that the model hugely depends on the communication channels and is prone to failure due to underlying technological failures. They remarked that the master node can become a single point of failure of the complete system. In [56] Ma and Gu denounced these points claiming that these are not "significant" limitations and also not technically a problem. However, according to them, the problem with MapReduce is that it's "one-way scalable", which means that by-design it allows to scale up to process large datasets, but in turn limits the capability to process smaller data items. Some other limitations mentioned by the community [57], [58] in general are that it's only for batch processing and not for interactive, iterative, streaming or incremental processing. It is also not suitable when there are too many keys because then sorting takes too long. MapReduce is not considered to be a good solution if the values depend on each other from one step to another. However, these limitations do not apply to our work, because in our case, we have clearly separated data set available for batch processing, where the values from one frame do not depend on the other.

More recently, researchers are looking towards Apache Spark [59] (initial release in 2014) as a fast, easy-to-use and realtime alternative to MapReduce; however for our scenario, MapReduce suits better because the high speed of Spark is relevant in scenarios when the "same" data has to be processed multiple times like in visualization applications. In our work the data is continuously changing and each frame needs to be processed only once. Similarly, our data is available as a batch and hence we do not need any real-time processing.

III. SIFT FOR FEATURE IDENTIFICATION

One of the first steps to computer vision applications is to extract the representative features and match them against a model. Some earlier work e.g. [60]–[62] proposed to use raw pixels for an effective initial feature representation for learning. However, this requires the overall number of pixels to be small, which is rarely the case in current imaging scenarios. E.g. in our dataset, each image is 1920 x 1080. Including the multi-camera views, the overall number of pixels becomes overwhelmingly large and hence computationally expensive. Most of the current work adopts the alternative approach to extract representative features for compact and effective representation. In the remainder of this section, we present an overview to the Scale Invariant Feature Transform – SIFT.

Though SIFT is one of the oldest techniques, it is the most reliable one that is still being used in most computer vision algorithm for object detection and tracking, robotic mapping and large scale image retrieval. The only disadvantage is that it takes more computation time and cannot be applied in real time environment. Aniruddha Acharya K and R. Venkatesh Babu presented a parallel implementation of SIFT on a GPU [63], in which they combined kernel optimization that has led to a significant improvement. In this paper we've overcome this problem by applying MapReduce for parallel feature extraction.

SIFT offers a number of advantages like scale-, rotation- and viewpoint- invariance and robustness to illumination changes. SIFT is extracted by progressively resizing and blurring out images to find features that are stable across a number of scales. Each set of images at a given size is referred to an octave. Blurring is performed by taking the convolution

of the Gaussian operator and the original image and it is applied to each pixel of an image. Mathematically, it is done using Eq (1).

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (1)$$

where

I and L →Original and Blurred Images

G → Gaussian Blur operator

x, y→ Location Coordinates

σ → Scale parameter or Amount of blur in a particular image.

(Typically $\sigma = \sqrt{2}$ *)*

** → Convolution operation in x and y*

The Gaussian blur operator is defined by Eq (2)

$$G(x, y, \sigma) = \frac{e^{-(x^2+y^2)/2\sigma^2}}{2\lambda\sigma^2} \quad (2)$$

Next it is needed to identify the keypoints which are maxima/ minima over multiple scales. For this purpose Difference of Gaussian (DoG) is computed. Specifically, a DoG image D(x, y, σ) is given by Eq (3).

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma) \quad (3)$$

where k_i and k_j represents two adjacent scales.

Next, each pixel is compared with its neighbors (i.e 8 comparison), its above image (i.e 9 comparisons) and its below image (i.e 9 comparison) making a total of 26 checks. It is considered as a keypoint if it is the extrema among all of its 26 neighbors. Next, to achieve rotation invariance the Gaussian-smoothed image *L(x, y, σ)* for all levels of scale σ are considered. For each and every pixel, which is around the region of a keypoint, the magnitude and angle is calculated using Eqs. (4) & (5) in the Gaussian-blurred image L.

$$m(x, y) = \sqrt{(L_{(x+1,y)} - L_{(x-1,y)})^2 + (L_{(x,y+1)} - L_{(x,y-1)})^2} \quad (4)$$

and

$$\theta(x, y) = \tan^{-1} \frac{(L_{(x,y+1)} - L_{(x,y-1)})}{(L_{(x+1,y)} - L_{(x-1,y)})} \quad (5)$$

where

m(x, y) → the gradient magnitude and *θ(x, y)* → orientation

Finally a descriptor vector for each keypoint is computed. This step is performed on the image closest in scale to the keypoint's scale. For this purpose a 16 x 16 pixel region is considered around each keypoint and is sub divided into 4 x 4 sub regions. A set of orientation histogram is created on 4 x 4 subregions with 8 bins each resulting in a descriptor 128 elements.

Consider Figure 2 that represents the SIFT keypoints for an image.

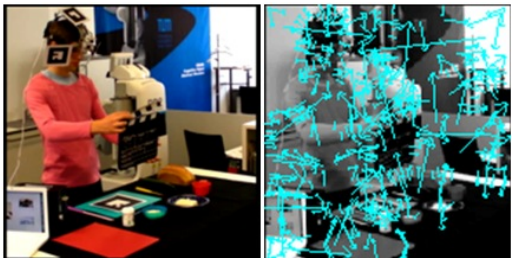

Fig. 2: SIFT points calculated for a sample image

IV. DISTRIBUTED COMPUTING PARADIGMS

Cluster-, grid- and cloud- computing are the most widely used distributed computing paradigms [64]. A number of attempts have been made to define the three from a number of perspectives e.g. [64]–[67]. Clouds are a collection of low- and high- end computers supporting virtualized nodes, which are provisioned on-demand and accessible as composable service via Web Service technologies such as Simple Object Access Protocol (SOAP) [68] or Representational State Transfer (REST) [69]. Clusters are typically under a single administrative domain and this distinguishes them from grids, which are spread geographically and are administered by multiple management policies and goals. Also clusters focus on enhancing overall system performance, while grids enhance application performance depending on the end-users' Quality-of-Service (QoS) requirements. A comparison of different distributed computing systems is given in Figure 3.

V. MAPREDUCE FOR COMPUTER VISION APPLICATIONS

Many image and video processing tasks typically comprise of a basic set of computation steps applied to a large amount of data. For example, a typical 3-minute video sequence shot @60 frames per second (FPS) results in over 10K images, with the number growing drastically as we add multiple viewpoints (cameras), longer sequences or higher frame rate etc. As shown in Figure 1, there are certain pre-processing and feature identification steps which need to be applied to each frame independently, before a model could be generated for the actual application. MapReduce supports scenarios like these where the same computation is desired on a stream of independent datapoints. Hence, the problem is divided into "mappers" that perform the feature extraction on the images in parallel; and "reducers" that aggregates the results.

MapReduce stands on the key-value pairs as the basic data structure. Keys and values can be as simple as primitives data types such as integers, floating points, strings, and even raw bytes, or they may be more complex structures like tuples, lists etc [70]. For our work, they are the image descriptors.

As described above, the programmer has to define a Map and a Reduce function which typically take the form as shown in Equations (6) and (7) [70]:

$$map : (k_1, v_1) \rightarrow [(k_2, v_2)] \quad (6)$$

$$reduce : (k_2, [v_2]) \rightarrow [(k_3, v_3)] \quad (7)$$

where, [...] represent a list of elements. The mapper is applied to each input data point (images in our case) and generates an arbitrary number of intermediate key-value pairs (SIFT points). This is followed by an implicit "ordering" or "grouping" phase to sort the keys. This is important for handling the distribution of data across multiple machines. Finally, the reducer is applied to all the values with the common intermediate key to generate the final key-value pairs.

Consider Figure 4 where we show MapReduce applied to parallelize our scenario. It is important to note that most of the subsequent steps in image processing applications are

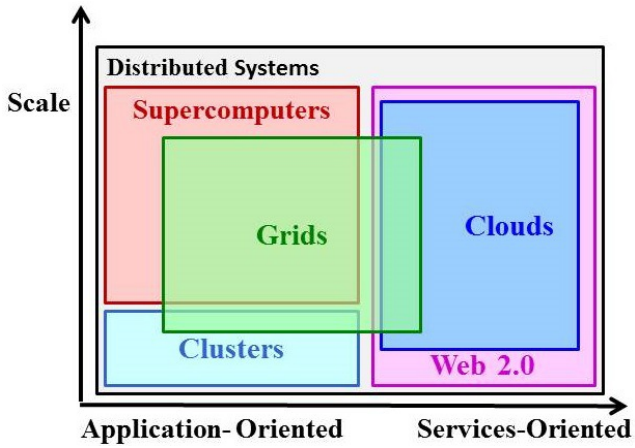

Fig. 3: Comparison of Different Distributed Architectures [45]

dependent on the initial time-intensive step of feature extraction. The number of datapoints depend on SIFT feature in each image, the number of images in each stream and the number of streams as shown in Eq. (8). Simple maths show that this number is quite high for any typical scene and rises quickly if we increase number of streams and/or images in any stream. In our work, we've exploited the parallelism while extracting features. Each mapper takes an image one-after-another as input and returns a set of SIFT-descriptors alongwith the location. The reducer using the function UPDATEMODEL aggregates these points from multiple streams and prepares them for subsequent steps. Consider Algorithm 1 that shows the algorithm for our scenario.

$$Datapoints = \sum_{k=0}^{n} \sum_{j=0}^{m} \sum_{i=0}^{l} (Feature_{ijk}, loc_{ijk}) \qquad (8)$$

where,
l: no. of features in an image,
m: no. of frames in a camera stream and
n: no. of camera streams.

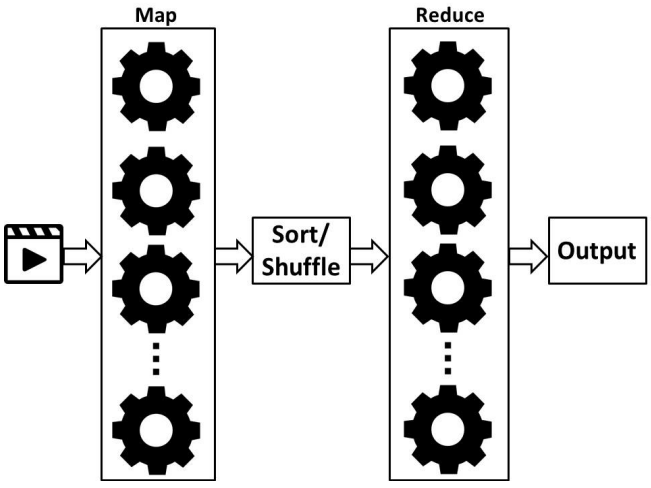

Fig. 4: MapReduce for fast Feature Identification

Algorithm 1 Algorithm for computing feature identification using MapReduce. The UPDATEMODEL function combines the individual features from all images for subsequent processing

```
1: Class Map
2:   method MAP (imageID d, image i)
3:     [<f, l>] ← ComputeFeature(image i)
4:     EMIT(tuple<imageID d>,tuple<feature[f],location[l]>)
5: Class Reduce
6:   method Reduce(tuple< image d>, tuples[t1, t2, ...])
7:     M←INITMODEL()
8:     for all <image d> ϵ tupleID do:
9:       for all tuple<feature[f], location[l]> ϵ features do:
10:      UPDATEMODEL (imageID d, feature[f],location[l],
          model M)
11:      YIELD (id m, Model M)
```

VI. EXPERIMENTS AND OBSERVATIONS

We considered the sandwich making scenario of the kitchen dataset from experiments conducted at the Technische Universität München (TUM) [71]. The sandwich making experiment comprises of 10 subjects, each having 18 iterations. Each set lasts for approximately 3 minutes, which implies a video of 570 minutes. This implies a dataset of 1.94m images @ 60 FPS. As described earlier, extracting features from this huge dataset is a pre-requisite, before a model can be generated and trained. This is a time-consuming step and we used the Amazon cloud to run Elastic MapReduce (EMR) to manage the high computation cost for the huge number of images. Using MapReduce allowed us to manage the high computation cost resulting due to the multiple input streams. We explored HIPI [72] - a Hadoop Image Processing Interface designed specifically for image-based MapReduce tasks and supports computer vision algorithms, however, we decided not to go for it as the technology is not yet mature, and hence there is a lack of benchmarks and expertise [73]. Also, HIPI is not suitable for such a huge dataset [74]. SIFT computation on local machine (2.5 GHz, dual-core, 8 GiB) took 2.087 sec/image totalling to an estimated 46.9576 days for the complete dataset of 1.94m images. Next, we extracted the SIFT features using parallel architecture. We considered using cluster, grid and cloud for our experiments. As shown in Table I, we analysed the three on a number of properties [64] and found that using a cloud will suit our purpose the best. However, we also executed the MapReduce on a local, non-cloud cluster to have a direct comparison. We ran the MapReduce on an 8-node local cluster with same configuration. It took the cluster 35.218 hours i.e. 1.46 days for execution, with a speedup of 32. However, we noticed that setting up our own cluster was restrictive because we could only expand to a certain number of devices. Also, there was a complicated and error-prone process of installation of distributed processing framework (we used Yahoo's Hadoop), configuration of nodes, and using custom libraries (we used mrjob) for execution. We realized that the effort required to setup our own cluster was more than the benefits it brought. Hence, we moved to Amazon Cloud and used Hadoop 1.0.3 to

TABLE 1: Comparison of Cluster, Grid and Cloud. An adaption of [64]

Characteristics	Systems			Preferred
	Clusters	**Grids**	**Clouds**	
Population	Commodity computers	High-end computers (servers clusters)	Commodity computers and high end servers and network attached storage	Cluster, Cloud
Size/scalability	100s	1000s	100s – 1000s	Cluster, Cloud
Node Operating System (OS)	One of the standard OSs (Linux/ Windows)	Any standard OS (dominated by Unix)	A hypervisor (VM) on which multiple Oss run	Cluster, Cloud, Grid
Ownership	Single	Multiple	Single	Cluster, Cloud, Grid
Interconnection network/ speed	Dedicated, high-end with low latency and high bandwidth	Mostly Internet with high latency and low bandwidth	Dedicated, high-end with low latency and high bandwidth	Cluster, Cloud, Grid
User management	Centralized	Decentralized and also Virtual Organization based	Centralized or can be delegated to third party	Cloud
Resource management	Centralized	Distributed	Centralized/ distributed	Cloud
Allocation/ scheduling	Centralized	Decentralized	Centralized/ decentralized	Cloud
Standards/ inter-operability	Virtual Interface Architecture	Open grid forum standard	Web Services (SOAP and REST)	Cloud
Single system image	Yes	No	Yes but optional	Cluster, Cloud
Capacity	Stable and guaranteed	Varies, but high	Provisioned on demand	Cloud
Failure management (self healing)	Limited (often failed tasks/ applications are restarted)	Limited (often failed tasks/ applications are restarted)	Strong support for fail over and content replication. VMs can be easily migrated from one node to another	Cloud
Pricing of services	Limited, not open market	Dominated by public good or privately assigned	Utility pricing, discounted for larger customers	Cloud
Internetworking	Multi-clustering within an organization	Limited adoption, but being explored through research efforts	High potential, third party solution providers can loosely tie together services of different clouds	Cloud
Application drivers	Science, business, enterprise	Collaborative scientific and high throughput applications	Dynamically provisioned legacy and web applications, content deliver	Cluster, cloud, grid
Potential building for 3rd party or value added solutions	Limited due to rigid architecture	Limited due to strong orientation for scientific computing	High - can create new services by dynamically provisioning compute, storage and application services	Cloud

run Elastic MapReduce. It was easier and cheaper to manage as all the installation and configuration load was taken away from us. Our final cluster comprised of 20 m4.large machines with the same configuration i.e 2.5 GHz, 2 vCPU, 6.5 ECU, 8 GiB on linux. We used spot instances and they provided a reasonable trade-off between computation capacity and price [75]. Using this system, the same calculation was done in 14.08 hours i.e. 0.586 days, thus giving us a speedup of 80 times. Consider Figure 5 that shows a comparison of time consumed on local machine versus that on cluster and cloud.

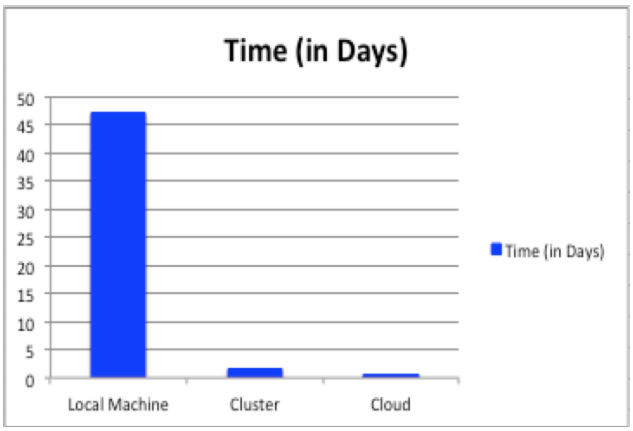

Fig. 5: Execution Time Comparison

VII. CONCLUSION

Feature extraction and identification is one of the prerequisite of any computer vision application. It is a time consuming step and all the following steps are dependent on it. In this paper we used MapReduce on Amazon cloud to parallelize the computation and showed that a high speedup can be obtained without dealing with the complexities of parallel computing and programming. We considered a huge sandwich making dataset of 1.94 m images and used SIFT descriptor as the feature. On our local machine, it took an estimated 46.9576 days to process the complete data. However, with our proposed approach of using EMR on Amazon cloud, the same was done in 0.586 days, thus resulting in a speedup of 80. In the future, we want to explore the effect of applying MapReduce in other phases of computer vision process like image alignment in case of multiple camera streams.

REFERENCES

[1] T. Kobayashi, "BFO Meets HOG: Feature Extraction Based on Histograms of Oriented p.d.f. Gradients for Image Classification," in Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, June 2013, pp. 747–754.

[2] C. T. Dadi El Wardani, Daoudi El Mostafa, "Improving 3D Shape Retrieval Methods based on Bag-ofFeature Approach by using Local Codebooks," International Journal of future Generation Communication and Networking, vol. 5 (4), pp. 29 – 38, 2012.

[3] A.Srinagesh, K.Aravinda, G. Varma, A.Govardhan, and M. SreeLatha, "A Modified Shape Feature Extraction Technique For Image

Retrieval," International Journal of Emerging Science and Engineering (IJESE), vol. 1 (8), pp. 9 – 13, June 2013.
[4] H. Moravec, "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," in Tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University, September 1980, no. CMU-RI-TR-80-03.
[5] C. Harris and M. Stephens, "A combined corner and edge detector," in Proc. of Fourth Alvey Vision Conference, 1988, pp. 147 – 151.
[6] Y. Xingfang, H. Yumei, and L. Yan, "An improved SUSAN corner detection algorithm based on adaptive threshold," in Signal Processing Systems (ICSPS), 2010 2nd International Conference on, vol. 2, July 2010, pp. V2–613–V2–616.
[7] S. M. Smith and J. M. Brady, "SUSAN - A New Approach to Low Level Image Processing," International Journal of Computer Vision, vol. 23, no. 1, pp. 45–78, 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1007963824710
[8] A. Sotiras, C. Davatzikos, and N. Paragios, "Deformable Medical Image Registration: A Survey," IEEE Transactions on Medical Imaging, vol. 32, no. 7, pp. 1153–1190, July 2013.
[9] E. Trucco and K. Plakas, "Video Tracking: A Concise Survey," IEEE Journal of Oceanic Engineering, vol. 31, no. 2, pp. 520–529, April 2006.
[10] R. Szeliski, "Image mosaicing for tele-reality applications," in Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on, Dec 1994, pp. 44–53.
[11] Y. Xiong and K. Pulli, "Fast panorama stitching for high-quality panoramic images on mobile phones," IEEE Transactions on Consumer Electronics, vol. 56, no. 2, pp. 298–306, May 2010.
[12] M. Brown and D. G. Lowe, "Automatic Panoramic Image Stitching using Invariant Features," International Journal of Computer Vision, vol. 74, no. 1, pp. 59–73, 2007. [Online]. Available: http://dx.doi.org/10.1007/s11263-006-0002-3
[13] S.-W. Ha and Y.-H. Moon, "Multiple Object Tracking Using SIFT Features and Location Matching," International Journal of Smart Home, vol. 5 (4), pp. 17 – 26, October 2011.
[14] F. Arman and J. K. Aggarwal, "Model-based object recognition in denserange imagesa review," ACM Computing Surveys (CSUR), vol. 25 (1), pp. 5–43, March 1993.
[15] J. Xiao and M. Shah, "Two-frame wide baseline matching," in Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, Oct 2003, pp. 603–609 vol.1.
[16] T. Tuytelaars and L. Van Gool, "Matching Widely Separated Views Based on Affine Invariant Regions," International Journal of Computer Vision, vol. 59, no. 1, pp. 61–85, 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000020671.28016.e8
[17] J. Matas, O. Chum, M. Urban, and T. Pajdl, "Robust wide-baseline stereo from maximally stable extremal regions," Image and Vision Computing, vol. 22(10), pp. 761 – 767, Sept 2004
[18] T. Kadir, A. Zisserman, and M. Brady, An Affine Invariant Salient Region Detector. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 228–241. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24670-1_18
[19] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," Int. J. Comput. Vision, vol. 60, no. 2, pp. 91–110, Nov. 2004.[Online].Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94
[20] T. Lindeberg, "Scale Invariant Feature Transform," Scholarpedia, vol. 7 No. (5), no. 10491, 2012.
[21] D. G. Lowe, "Object Recognition from Local Scale Invariant Features," in Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2, ser. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–. [Online]. Available: http://dl.acm.org/citation.cfm?id=850924.851523
[22] H. K. P. Pandya, J. Senjalia, "Evaluating the object recognition in realtime process," Nirma University International Conference Engineering (NUiCONE), pp. 1 – 6, 2013.
[23] K. Aniruddha Acharya and R. Venkatesh Babu, "Speeding up SIFT using GPU," in Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2013 Fourth National Conference on, Dec 2013,pp. 1–4.
[24] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-Up Robust Features (SURF)," Computer Vision and Image Understanding, pp. 346– 359, 2008
[25] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 27, no. 10, pp. 1615–1630, 2005. [Online]. Available: http://lear.inrialpes.fr/pubs/2005/MS05
[26] M. Sarfraz and O. Hellwich, "On Head Pose Estimation in Face Recognition," in Computer Vision and Computer Graphics. Theory and Applications, ser. Communications in Computer and Information Science, A. Ranchordas, H. Arajo, J. Pereira, and J. Braz, Eds. Springer Berlin Heidelberg, 2009, vol. 24, pp. 162–175. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10226-4_13
[27] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a Local Binary Descriptor Very Fast," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 7, pp. 1281–1298, July 2012.
[28] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary Robust Invariant Scalable Keypoints," in Computer Vision (ICCV), 2011 IEEE International Conference on, Nov 2011, pp. 2548–2555.
[29] P. V. Alexandre Alahi, Raphael Ortiz, "FREAK: Fast Retina Keypoint," in Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), ser. CVPR '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 510–517. [Online]. Available: http://dl.acm.org/citation.cfm?id=2354409.2354903
[30] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in In European Conference on Computer Vision, 2006, pp. 430 – 443.
[31] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in Computer Vision (ICCV), 2011 IEEE International Conference on, Nov 2011, pp. 2564–2571.
[32] K. Potisepp, "Large-scale image processing using MapReduce," Master's thesis, Faculty of Mathematics and Computer Science Institute of Computer Science, 2013.
[33] D. Moise, D. Shestakov, G. Gudmundsson, and L. Amsaleg, "Indexing and searching 100m images with MapReduce," in Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval (ICMR), ser. ICMR '13. New York, NY, USA: ACM, 2013, pp.17–24.[Online].Available: http://doi.acm.org/10.1145/2461466.2461470
[34] M. Yamamoto and K. Kaneko, "Parallel image database processing with MapReduce and performance evaluation in pseudo distributed mode," International Journal of Electronic Commerce Studies, vol. 3 Issue (2), pp. 211 – 228, 2012.
[35] V. A. Natarajan, S. Jothilakshmi, and V. N. Gudivada, "Scalable Traffic Video Analytics using Hadoop MapReduce," The First International Conference on Big Data, Small Data, Linked Data and Open Data, pp. 11 – 15, April 2015.
[36] Y. J. Yu and K. Hu, "Mono Image Based Object Recognition With MapReduce," 2014.
[37] F. Seinstra and J. Geusebroek, "Color-Based Object Recognition on a Grid," 2006.
[38] F. J. Seinstra and J. M. Geusebroek, "A Demonstration of Color-Based Object Recognition on a Grid."
[39] F. J. Seinstra and J.-M. Geusebroek, "Color-Based Object Recognition by a Grid-Connected Robot Dog."
[40] W. J. Beksi, J. Spruth, and N. Papanikolopoulos, "CORE: A Cloud based Object Recognition Engine for robotics," in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sept 28 Sept 2015 - 02 Oct 2015, pp. 4512–4517.
[41] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, "Cloud-based robot grasping with the google object recognition engine," in IEEE Intl Conf. on Robotics and Automation, 2013.
[42] "Cloud Storage - A Powerful, Simple and Cost Effective Object Storage Service." [Online]. Available: https://cloud.google.com/storage/
[43] W. Garage, "Personal Robot 2." [Online]. Available: http://www.willowgarage.com
[44] A. K. Paul and J. S. Park, "Multiclass object recognition using smart phone and cloud computing for augmented reality and video surveillance applications," in Informatics, Electronics Vision (ICIEV), 2013 International Conference on, May 17 - 18 May 2013, pp. 1–6.

[45] D. Lorencik, M. Tarhanicova, and P. Sincak, Robot Intelligence Technology and Applications 2: Results from the 2nd International Conference on Robot Intelligence Technology and Applications. Cham: Springer International Publishing, 2014, ch. Cloud-Based Object Recognition: A System Proposal, pp. 707–715. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-05582-4 61

[46] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Sixth Symposium on Operating System Design and Implementation (OSDI), December 2004.

[47] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," Journal of the ACM (JACM), vol. 27 Isuue (4), no. 4, pp. 831–838, Oct. October 1980. [Online]. Available: http://doi.acm.org/10.1145/322217.322232

[48] K.-Y. Liu, S.-Q. Li, L. Tang, L. Wang, and W. Liu, "Fast face tracking using parallel particle filter algorithm," in Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on, June 28 June - 3 July 2009, pp. 1302–1305.

[49] Y. Li, D. J. Crandall, and D. P. Huttenlocher, "Landmark classification in large-scale image collections," in Computer Vision, 2009 IEEE 12th International Conference on, Sept 2009, pp. 1957–1964.

[50] K. Lyndon, S. Malcolm, and W. Kilian, "Reliable Tags Using Image Similarity: Mining Specificity and Expertise from Largescale Multimedia Databases," in Proceedings of the 1st Workshop on Web-scale Multimedia Corpus, ser. WSMC '09. New York, NY, USA: ACM, 2009, pp. 17–24. [Online]. Available: http: //doi.acm.org/10.1145/1631135.1631139

[51] D. Markonis, R. Schaer, I. Eggel, H. Mller, and A. Depeursinge, "Using MapReduce for Large-Scale Medical Image Analysis," in Healthcare Informatics, Imaging and Systems Biology (HISB), 2012 IEEE Second International Conference on, Sept 2012.

[52] W. Han, Y. Kang, Y. Chen, and X. Zhang, "A MapReduce Approach for SIFT Feature Extraction," in Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on, Dec 2013, pp. 465–469.

[53] D. DeWitt and M. Stonebraker, "MapReduce: A major step backwards," 2010. [Online]. Available: http://databasecolumn.vertica.com/database-innovation/mapreduce-a-major-step-backwards/

[54] Y. Yu, M. Isard, D. Fetterly, M. Budiu, lfar Erlingsson, P. K. Gunda, and J. Currey, "DryadLINQ: a system for general-purpose distributed data parallel computing using a high-level language," Proceeding OSDI'08 Proceedings of the 8th USENIX conference on Operating systems design and implementation, pp. 1 – 14, 2008.

[55] C.-T. Chu, S. K. Kim, Y. an Lin, Y. Yu, G. Bradski, K. Olukotun, and A. Y. Ng, "Map-Reduce for Machine Learning on Multicore," in Advances in Neural Information Processing Systems 19, B. Schoelkopf, J. C. Platt, and T. Hoffman, Eds. MIT Press, 2007, pp. 281–288. [Online]. Available: http://papers.nips.cc/paper/3150-map-reduce-for-machine-learning-on-multicore.pdf

[56] Z. Ma and L. Gu, "The Limitation of MapReduce: A Probing Case and a Lightweight Solution," The First International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2010).

[57] [Online]. Available: https://www.quora.com/What-are-some-limitations-of-MapReduce

[58] [Online].Available: http://stackoverflow.com/questions/18585839/what-are-the-disadvantages-of-mapreduce

[59] [Online]. Available: http://spark.apache.org/

[60] G. E. Hinton, "Learning multiple layers of representation," TRENDS in Cognitive Sciences, vol. 11 No, (10), pp. 428 – 434, 2007.

[61] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[62] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, Computer Vision – ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part VI. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ch. Convolutional Learning of Spatio-temporal Features, pp. 140 – 153. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15567-3 11

[63] K. A. Acharya and R. V. Babu, "Speeding up SIFT using GPU," Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), pp. 1 – 4, 2013.

[64] R. Buyya, C. S. Yeo, S. Venugopal, and J. B. and Ivona Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25 issue(6), pp. 599 – 616, June 2009.

[65] J. Geelan, "Twenty-one experts define cloud computing," Cloud Expo Journal, 24 January 2009.

[66] G. F. Pfister, In Search of Clusters (2nd Ed.). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998.

[67] R. Buyya, High Performance Cluster Computing: Architectures and Systems. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[68] [Online]. Available: https://en.wikipedia.org/wiki/SOAP

[69] [Online].Available: https://en.wikipedia.org/wiki/Representational_state_transfer

[70] J. Lin and C. Dyer, Data-Intensive Text Processing with MapReduce. Morgan & Claypool Publishers, 2010.

[71] "TUM cooking datasets." [Online]. Available: http://web.ics.ei.tum.de/_karinne/Dataset/dataset.html

[72] C. Sweeney et al "HIPI: A Hadoop Image Processing Interface for Image- based MapReduce Tasks," University of Virginia, Undergraduate Thesis, 2011.

[73] S. M. Banaei and H. K. Moghaddam, "Hadoop and Its Role in Modern Image Processing," Open Journal of Marine Science, vol. 4, issue (4) October 2014. http://dx.doi.org/10.4236/ojms.2014.44022

[74] B. Kanoongo, P Jagani and C. Bhadane, "Distinction of Discrete Transformations Applied to Hadoop's MapReduce," International Journal of Computer Applications, vol. 104, issue (10), pp. 0975 – 8887, October 2014.

[75] "Amazon EC2 pricing." [Online]. Available: https://aws.amazon.com/ec2/pricing/