Content based Image Retrieval with Graphical Processing Unit

Bhavneet Kaur¹ and Sonika Jindal²

¹ Mtech Student, Computer Science and Engineering, SBS State Technical Campus, Ferozepur, Punjab, INDIA
Email: sethi.bhavneet37@gmail.com

² Assistant Professor, Computer Science and Engineering, SBS State Technical Campus, Ferozepur, Punjab, INDIA
Email: sonikamanoj@gmail.com

Abstract— CBIR is the method of searching the digital images from an image database. “Content-based” means that the search analyzes the contents of the image rather than the metadata such as colours, shapes, textures, or any other information that can be derived from the image itself. The GPU is a powerful graphics engine and a highly parallel programmable processor having better efficiency and high speed that overshadows CPU. It is used in high performance computing system. The implementation of GPU can be done with CUDA C. Due to its highly parallel structure it is used in a number of real time applications like image processing, computational fluid mechanics, medical imaging etc. Graphical Processors Units (GPU) is more common in most image processing applications due to multithread execution of algorithms, programmability and low cost. In this paper, we are explaining the parallel implementation of CBIR with GPU. We have shown various stages of CBIR with GPU results into better performance as well as speed ups. We have given a review of various techniques that can be practised for high performance CBIR stages with Graphics Processing Units.

I. INTRODUCTION

GPU is Graphics Processing Unit used for high performance parallel computing. It is a processor that over the past few years has evolved from a fixed-function special-purpose processor into a full-fledged parallel programmable processor with additional fixed-function special-purpose functionality. More than ever, the programmable aspects of the processor have taken centre stage [1]. GPU computing is the use of GPU together with CPU (Central Processing Unit) to accelerate general-purpose scientific and engineering applications. The GPU is a processor with ample computational resources. GPU computing is the use of a GPU together with a CPU to accelerate general-purpose scientific and engineering applications [1]. Fig.1 shows the architecture of GPU where CPU sends tasks and data to GPU whereas GPU performs computations on data and sends back results to CPU. The CPU controls all the computations. So, in context of GPU computing:

- GPU code is called DEVICE code.
- CPU code is called HOST code.

GPU is a type of highly parallel, multi-threaded and multi-core processor. A list of geometric primitives like triangle in 3-D world co-ordinate system, is given as an input to the GPU which are shaded and mapped onto

DOI: 02.ITC.2014.5.91
© Association of Computer Electronics and Electrical Engineers, 2014
the screen and further assembled to develop a complete image or a picture [1]. The GPU graphics pipeline undergoes various steps like vertex operations, Primitive assembly, Rasterization, Fragment operations and Composition [1]. From this pipeline structure, GPUs have become further programmable. Purpose GPU is quite suitable for computing intensive data parallel. In Jun. 2007 NVIDIA released CUDA and in Dec 2008, Khronos Group released OpenCL1. In Aug. 2009, AMD launched ATI Stream SDK v2.0 Beta which supported X86 processor. Open CL is an open standard for many processors [2]. Modern GPUs contain hundreds of processing units, capable of achieving up to 1 TFLOPS for single-precision (SP) arithmetic, and over 80 GFLOPS for double-precision (DP) calculations. Recent High Performance Computing (HPC) optimized GPUs contain up to 4GB of on board memory and 100GB/sec [3]. Due to its parallel architecture and high performance of floating point and memory operations GPU is well suited for many same scientific and engineering applications that occupy HPC clusters, leading to their incorporation as HPC accelerators [3]. So, they can reduce space, power, and cooling demands, and reduce the number of operating system images that must be managed relative to traditional CPU-only clusters of similar aggregate computational capability [3]. GPU has its amazing computational capabilities and functionalities that extends its applications to the field of non-graphics computations and such a type of GPU is known as General Purpose GPU [4]. Due to their cost performance and evolution speed they are becoming significant [4].

A. CUDA

CUDA is Compute Unified Design Architecture. It was introduced by NVIDIA in November 2006. It allows software developers to access GPUs through standard programming language, ‘C’. For CUDA (C with NVIDIA extensions and certain restrictions). A compiled CUDA program can execute on any number of
processor cores. CUDA C is the language specifically designed to provide general purpose computing on GPU. CUDA C adds the global qualifier to standard C. This mechanism alerts the compiler that a function should be compiled to run on a device instead of the host. In this simple example, nvcc gives the function kernel ( ) to the compiler that handles device code, and it feeds main ( ) to the host compiler. It provides this language integration so that device function calls look very much like host function calls [5]. Fig. 2 shows the architecture of CUDA which is composed of Grids and Blocks. In the grid there are multiple threads of the main kernel running parallel and each grid composes a number of blocks that contains the threads and the shared memory [5]. The NVIDIA CUDA technology (developer.download.nvidia.com) is a fundamentally new computing architecture that enables the GPU to solve complex computational problems.

Fig. 2: Various Stages of CBIR

II. CONTENT BASED IMAGE RETRIEVAL AND GPU

Content Based Image Retrieval is based upon retrieving an image by comparing a query image with all the images in the database. Fig. 3 shows the certain steps that are followed to fetch an image from the database. Graphical Processors Units (GPU) is more common in most image processing applications due to multithread execution of algorithms, programmability and low cost. CBIR along with GPU finds its applications in medical field [7]. Medical imaging produces great amounts of data which, once investigated and classified, might be extremely valuable for future diagnoses. The image retrieval in medical applications (IRMA, irma-project.org) approach aims at providing a framework for medical CBIR applications including interfaces to PACS and hospital information systems (HIS) [6] where IRMA exactly addresses the Kilo- to Terabyte sized datasets challenge in medical image management and data mining. The main disadvantage of working with Giga- to Terabyte volume data is the runtime performance. This can be achieved by two methods [7]:

1. Parallel CPU-based programming on a single node with shared memory using threaded programming techniques like Open MP or Qt Threaded.
2. Parallel GPU-based programming on a single node with one GPU or multiple GPUs using programming languages for the massive parallel cores on the graphic card [8]. So, to have efficient algorithms one can use GPU with CUDA or Open CL.
3. Parallel programming on multiple nodes in a cluster of linked computers connected through a fast local area network (LAN), which is also referred to as Grid computing [9].

This paper gives an overview of literature showing various stages of CBIR where GPU can be used along with it so as to result in faster speed ups and high performance. Fig. 4 shows those stages that are reviewed in this work.

III. FEATURE EXTRACTION WITH GPU

In content-based image retrieval, images are automatically stored at index in feature database which describe the content of the image. The features are extracted and stored as feature vectors. The feature vectors of both the query image and the image database are compared and thus the required image is retrieved. Features may be colour, shape, contour, etc. [10] Q be the query image, T be the image database and t be the threshold distance. So, the distance between both the feature vectors is given as,

\[ D(\text{Feature}(Q), \text{Feature}(T)) \leq t. \]  (1)
Features can be extracted in parallel from the images with GPU using various techniques. Feature extraction can be performed by using GPU based KLT feature tracker and GPU based SIFT feature extraction algorithm. The KLT tracking algorithm computes displacement of features or interest points between consecutive video frames when the image brightness constancy constraint is satisfied and image motion is fairly small[12]. In its parallel implementation the GPU distributes its various steps in fragment programs.

![Fig.4: CBIR stages with GPU](image)

Where frame performs tracking using the image pyramids of multi-resolution and intensity. SIFT is scale invariant feature transform algorithm [11] which performs extraction of interest points invariant to translation, rotation, scaling and illumination changes in images. It constructs a Gaussian scale-space pyramid from the input image and also calculates the gradients and difference-of-Gaussian (DOG) images at these scales. Interest points are detected at the local extremes within the DOG scale space. With GPU, the construction of the Gaussian scale space pyramid is accelerated by using fragment programs for separable Gaussian convolution. These implementations are 1020 times faster than the corresponding optimized CPU counterparts and enable real-time processing of high resolution video [12]. We can also use the improved version of parallel SIFT algorithm that provides better performance on multi core platforms [13] and takes care of following:

1. Load Balancing.
2. Reducing Synchronization Overhead.
3. Removing False Sharing.
4. Applying Thread Affinity.

GPU-based KLT implementation tracks about a thousand features in real-time at 30 Hz on 1024x768 resolution video which is a 20 times improvement over the CPU. It works on both ATI and NVIDIA graphics cards. The GPU-based SIFT implementation works on NVIDIA cards and extracts about 800 features from 640x480 video at 10Hz which is approximately 10 times faster than an optimized CPU implementation [14].

There is another application of feature detection where eye blink detector works on very low contrast images acquired under near-infrared illumination with GPU[15]. Eye blinks are detected inside regions of interest that are aligned with the subjects eyes at initialization. Alignment is maintained through time by tracking SIFT feature points that are used to estimate the affine transformation between the initial face pose and the pose in subsequent frames. Eye blink detection obviously implies prior detection of the eyes in the image of the subjects face. Here also GPU based implementation of SIFT is used for tracking as provided in the library Open NVIDIA [16] openvidia.sourceforge.net.

Object detection is the ability to detect and localize objects within an image or a scene [18]. Here also the features for an object are needed to be extracted. One of the algorithm adopted for object detection is AdaBoost[17] which further can be run on Graphics Processing Units[18]. This particular system can be evaluated with two face-detection applications which are based on the boosted cascade of classifiers: Multiple Layers Face Detection (MLFD), and Single Layer Face Detection (SLFD)[18]. It can be observed that SLFD implementation on GPU performs up to nine times faster than its CPU counterpart. The MLFD, in the other hand, can be accelerated using the GPU and performs up to three times faster than the CPU[18]. In [19] it is referred further to a method where focus is on the nuclei detection on Hematoxilin eosin stained colon tissue sample images. It examines that how effectively the algorithms used during the process can be implemented to data parallel architectures, and is it worth using GPU (Graphic Processing Unit) instead of the CPU (Central Processing Unit).
There is another approach where feature vectors can be created by using SIFT and GPU. All the stages of SIFT are optimized for better GPU optimization and on comparing the entire algorithm against original CPU implementation and manually SSE version, it can be notified that this system is evaluated with two face-detection applications. Those applications are based on the boosted cascade of classifiers: Multiple Layers Face Detection (MLFD), and Single Layer Face Detection (SLFD). We show that the SLFD implementation on GPU performs up to nine times faster than its CPU counterpart. The MLFD, on the other hand, can be accelerated using the GPU and performs up to three times faster than the CPU [20].

One of the methods is online Feature Extraction method which is used in case of meta-search engine where we do not maintain any content-based index and so we need to extract the content features during the query processing [21]. Here MPEG-7 visual descriptors and so-called image feature signatures are used. Once a ranking of images is returned from the other search engines given a keyword query, a feature extraction on the top images is executed. Since the online feature extraction is needed to be performed faster and without delays so its GPU based implementation should be used.

Feature extraction also finds its application in medical images classification processes for early skin cancer detection where the images are classified based upon Haralick features and fractal geometry. The speed and efficiency of texture and fractal analysis can be improved by taking its parallel implementation on GPU. In order to parallelize the code and execute it in on the GPU, the previously used Matlab GPU routines are rewritten using Jacket which is a runtime platform [22] that helps to connect the M language to the GPU. It offers support for specific data types counterparts to CPU Matlab data types, and a set of GPU functions ranging from basic implementations to complex arithmetic or signal processing solving methods.

SURF algorithm is another approach which is a multi scale feature detector that comprises three fundamental steps [23]:
1. Feature detection in scale-space.
2. Orientation assignment.
3. Feature description.

SURF can also be speeded up by using it with GGPU [20] [14] and suSURF[24].

Another technique of feature extraction is GLCM and Haralick Texture features. GLCM is the Gray Level Co-occurrence Matrices which is a common way to extract the texture features. It contains the second-order statistical information of spatial relationship of the pixels of an image. Haralick texture features are extracted using these GLCMs. These techniques can be implemented faster by using different task and data parallelism [25]. The computation of both of these techniques can be accelerated using Graphics Processing Units (GPUs) for biological applications [25]. Since in biological applications, features are extracted from microscopy images of cells so, it takes several weeks because of processing a larger number of images. Therefore, GPUs are used due to its less development time and faster growth.

The implementation of feature extraction on GPU can also be done by mapping each sub-image to one block each image-block is processed by one thread. Each Image is two-dimensional space that can be mapped to CUDA threads. There are 16 blocks in grid and also 64 threads in each block. Therefore, feature extraction is performed in all image-blocks in parallel by CUDA threads [26].

IV. SIMILARITY MATCHING WITH GPU

In this phase the feature vectors of the query image as well as that of the image database is compared for their similarities and thus the best suitable match is searched for. When dealing with inverse problems such as de-noising or de-convolution of images, a similarity measure is needed to evaluate how well the estimate explains the observations. Graphical Processors Units (GPUs) play important role to speedup processing of database images matching algorithms because it has more inbuilt execution cores [27]. One of the techniques is where the feature space as well as similarity on this space is defined. The feature space is based on a global descriptor of the image in a multi scale transformed domain [28]. After decomposition into a Laplacian pyramid, the coefficients are arranged in intra-scale/inter-scale/inter-channel patches which reflect the dependencies of neighbouring coefficients in presence of specific structures or textures. At each scale, the probability density function (pdf) of these patches is used as a descriptor of the relevant information [28]. Because of the sparsity of the multi scale transform, the most significant patches, called Sparse Multi-scale Patches (SMP), describe efficiently these pdfs. The Kullback-Leibler divergence method is used for statistical measure which is based on the comparison of these probability density functions that quantifies the closeness between two probability density functions and already shown good performances in the context of image retrieval. The similarity measure is done by \( k_m \) nearest neighbour search. To speed up the computation
time, parallel implementation of the KNN search on a Graphic Processing Unit (GPU) using CUDA is developed and it is observed that the computation time for one similarity measure between two images required 0.2s on average [28]. The online image matching can be enhanced by using GPUs [29]. Here it is performed in two steps given two images I and J, then,

1. The GPU computes in parallel the distances for each pair of descriptors \((i, j) \in I \times J\).
2. Targeting to two best matches filtering, the GPU then looks for the two closest descriptors \(j_i\) and \(j_i'\) for each descriptor \(i\) using a so-called reduction operation.

On comparing GPU matching method to ANN and naive \(O(m^2)\) CPU matching it can be seen that the times for CPU method grows faster than GPU method due to cache effect that make GPU memory faster an CPU/GPU memory transfers to consume time[29]. Another fast searching approach is FATS i.e. Fast Active Tabu Search that runs on the GPU platform [30]. This method is quite efficient in solving the large-scale combinatorial optimization problems. The work is implemented in three steps as following:

1. A new entropy function (GEOMEN) i.e. Geometric Manifold Entropy is defined to describe the connection and relevance between images.
2. The retrieval is treated as searching for an ordered cycle in an image database.
3. Tabu search is a common solution to the optimization problems. However, picking up the best candidate in this method is very time consuming, especially for large scale data sets. That is why fats is used whose main advantage is that it is very efficient for the large-scale optimization problems. Besides, FATS also can be applied to other related combinatorial optimization problems. The use of GPU in the minimization yields a very considerable speedup.

The similarity matching can also be performed by EUCLIDEAN DISTANCE FORMULA and MAHALANOBIS DISTANCE in which a query Image is entered and the distance from its points are used to match with the points on other image to get the results in case of Euclidean distance. This process is changed a bit and repeated in the same way to match the query Image from the Database in case of the Mahalanobis Distance. Both of these distances find their application in CBMIR (Content Based Medical Image Retrieval). The distance formula can be implemented in NVIDIA’s GPU programming environment, CUDA v0.9 [31].

Another technique is multi view stereo (MVS) matching problem [32] is formulated as follows: Given \(N\) calibrated images \(I = I_0, I_1, ..., I_{N-1}\) and corresponding projection matrix \(P = P_0, P_1, ..., P_{N-1}\), find a set of 3D points \(X = X_0, X_1, ..., X_{M-1}\) where the projection of \(X_i\) to its supporting images preserves photo consistency. So, the local window around the projected position is well matched in terms of SAD or NCC metrics.

A. Similarity Search with KNN AND GPU

KNN (K Nearest Neighbour) search algorithm is a nonparametric method for classifying objects based on closest training examples in the feature space. It is also called the Brute Force Method. Here the function is only approximated locally and all computation is deferred until classification. GPU can accelerate the process of the KNN search using NVIDIA CUDA. KNN is also known as Exhaustive Search method. It is a problem encountered in many graphics and non-graphics applications. Let \(R = r_1, r_2, ..., r_n\) be the reference points and \(Q = q_1, q_2, ..., q_n\) be the query points in the same space. The \(k\) nearest neighbour search problem consists in searching the \(k\) nearest neighbours of each query point \(q_i\) in \(Q\) in the reference set \(R\) given a specific distance. The distance included in it may be Euclidean distance, Manhattan distance or Mahalanobis distance [33]. The major problem with this algorithm is its huge complexity that is \(O(n m d)\) for the \(nm\) distances computed and \(O(nm \log m)\) for the \(n\) sorts performed [33]. So, KNN algorithm is highly parallelizable with GPU where an attempt is made to reduce the computed distances.

Fig. 5 shows the KNN search for \(K=3\) where there is a set of some reference points in black dots and red cross is the query point. KNN can also be done for complex query set where coarser level of parallelism can be exploited [34]. Let \(Q\) be the rows holding the queries and \(u\) be the columns storing database. So, there matrix multiplication at the first step can be implemented efficiently using GPU [34]. The sorting phase can be done in three steps. First, the distance vector is evenly distributed to CUDA Block threads. The elements are aligned in round robin pattern and each thread keeps its own private results. In the second and third step the local heaps are reduced and it is performed once all the threads have finished LC indices elements solving the KNN queries [34].
There is another approach which uses the same Brute-Force algorithm with API CUDA and CUBLAS [35], where CUBLAS is the CUDA implementation of BLAS and improves the performance of the classical BLAS functions. BLAS is the extended library for matrix and vector operations. Spaghettis structure can be used for parallel platforms based on multi-core and many-core processors. These implementations can be evaluated by using two different databases, Spanish dictionary and colour histograms. When considering all the implementations, the brute force implementation included, the behaviour of the structure in both metric spaces is similar, obtaining results of speedup between 1.87 and 3.94 for multi-core implementation, and between 2.08 and 14.04 for the GPU based platform. Considering the best implementations we can obtain a maximum speed up of 3.17 for multi-core and 9.84 for GPU, both cases obtained for the Spanish dictionary, which uses a more expensive computational cost distance function [36].

Similarity matching of feature vectors can be performed with KNN and for good speed ups the implementation is done on Graphics Processor Units. The Brute Force method can be implemented in CUDA, MATLAB and KDT-C and it is shown that the implementation done in CUDA is 120 times faster than BF MATLAB, 100 times faster than BF-C, and 40 times faster than KDT-C [14]. LSH (Locality Sensitive Hashing) [37], uses hash functions to compute the distances between a given query point and a subset of the reference points, which also an alternative method used in KNN search for further application image processing field.

IV. INDEXING WITH GPU

In Content-based Image Retrieval (CBIR) systems, accurately ranking images is of great relevance, since users are interested in the returned images placed at the first positions, which usually are the most relevant ones. In general, CBIR systems consider only pair wise image analysis, that is, compute similarity measures considering only pairs of images, ignoring the rich information encoded in the relations among several images. On the other hand, the user perception usually considers the query specification and responses in a given context.

In CBIR the similar images are collected by considering its visual properties and ranked in decreasing order of similarity, according to a given image descriptor. An image content descriptor is characterized by [38]:

1. An extraction algorithm that encodes image features into feature vectors;
2. A similarity measure used to compare two images.

The similarity between two images is computed as a function of the distance of their feature vectors. In CBIR, the relationships among images, encoded in ranked lists and distances among images, can be used for extracting contextual information [38]. There is a ranking method where ranking aggregation is done which uses the re-ranking methods to combine CBIR descriptors. The post-processing of the distance/similarity scores, by taking into account the contextual information available in relationships among images in a given collection. The methods require no user intervention, training or labelled data, operating on an absolutely unsupervised way. After that the outputs of re-ranking and rank aggregation methods, are combined aiming at further improving the effectiveness of CBIR results. Finally, for efficient performance the ranking approach is implemented on GPU [38].

In [39] visual word based recognition scheme is done and extends it by adding geographical dimensions to the visual words and use them to index 2D locations in a map grid. An indexing friendly scoring system is defined to measure the similarity of query and database images which represent unit tiles of the complete map. The implemented scoring algorithm can efficiently give the matching scores between a query image and
all possible database images. Upon searching a new approximately orthogonal image, a set of scaling and rotations are first selected, and the visual words are transformed and matched against the database. The best locations along with scales and rotations are determined from the query results of the different set of transformed visual words. Experiments show a high success rate and high speed in searching map databases for aerial images from different datasets. To extract SIFT features for satellite map database, the huge map image need to be divided into small pieces on which feature detection is run. As it is not feasible to detect features for the entire map at once. Enough overlap between sub-images is very important for keeping the features that are close the boundary. Otherwise, features of large scale on the boundary will be lost. Here, GPU-based SIFT implementation is used to speed up the processing [14]. This method is used for information mining from map databases, for example, searching for interesting patterns on a map.

There is another approach where the evaluation of effectiveness of different feature vectors for 2D photo organization is done [40]. A performance metric is proposed to measure how well photos with similar visual contents are grouped together on the 2D canvas. The organization of photos on 2D virtual canvas according to their similarities is done. Photo content is first analyzed and then transformed into a feature vector for arrangement processing. A Self Organizing Map (SOM) is then used to find the optimal location for each photo so that the ones with similar feature vectors are closer to each other. To speed up the SOM training processing, the algorithm is implemented on the Graphics Processing Unit (GPU) of programmable graphics Hardware [40]. In the end, the users can interactively organize thousands of photos and browse them through intuitive operations, such as pan and zoom.

The process of indexed object recognition can be speeded up by exploiting the inherent data parallelism [41]. This can be achieved by utilizing the Graphics Processor Units (GPU). New generation GPUs have a many-core architecture and support running thousands of threads in parallel. Many algorithms in pattern recognition and machine learning have been speeded-up using GPUs in the recent past, such as Neural networks enisen.dk/fann/html latest/files2/gputxt.html, SVMs, Decision trees, etc. GPUs have allowed the classification on large data sets to become feasible [41].

V. CONCLUSIONS

We have given a review of different techniques for various stages in CBIR that can be performed along with the Gpus. GPU enhance the speed as well as the performance of the method adopted for image retrieval. Graphical Processors Units (GPU) is more common in most image processing applications due to multithread execution of algorithms, programmability and low cost.

ACKNOWLEDGMENT

The success of this work would not have been possible without the encouragement and guidance of Mr. Sarabjot Singh, Assistant Professor, Shaheed Bhagat Singh State Technical Campus, Ferozepur, Punjab, INDIA. A special word of thanks must also be given to Dr. Satvir Singh, Associate Professor, Shaheed Bhagat Singh State Technical Campus, Ferozepur, Punjab, INDIA for his special contribution in context to research ideologies.

REFERENCES

[22] D. Ramona, “Matlab medical images classification on graphics processors”.
[37] D. Ramona, “Matlab medical images classification on graphics processors”.
[44] D. Ramona, “Matlab medical images classification on graphics processors”.


[38] D. C. G. Pedronette, R. d. S. Torres, “Exploiting contextual information in image retrieval tasks”.

