Journal of AI and Data Mining
Vol 7, No 1, 2019, 97-108
DOI: 10.22044/JADM.2017.4152.1503

Parallel Spatial Pyramid Match Kernel Algorithm for Object Recognition using a Cluster of Computers

A. AsilianBidgoli[1,2*], H. Ebrahimpour-Komleh[1], M. Askari[1] and SJ. Mousavirad[1]

1. Department of Computer Engineering, University of Kashan, Kashan, Iran.
2. Faculty of Electronic & Computer Engineering, Pooyesh Higher Education Institute, Qom, Iran.

Received 25 April 2016; Revised 12 June 2017; Accepted 19 July 2017
**Corresponding author: asilian@gmail.com (A. AsilianBidgoli).*

Abstract
This paper parallelizes the spatial pyramid match kernel (SPK) implementation. In the recent years, SPK has been one of the most usable kernel methods along with support vector machine classifier with high accuracy in object recognition. The MATLAB parallel computing toolbox is used to parallelize SPK. In this implementation, the MATLAB Message Passing Interface (MPI) functions and the features included in the toolbox help us obtain a good performance by the two schemes task-parallelization and data-parallelization models. The parallel SPK algorithm runs over a cluster of computers and achieves less run time. The observed speed-up depends on the number of CPUs and their cores. A speed-up value equal to 13 was obtained for a configuration with up to 5 Quad processors.

Keywords: *Object Recognition, Spatial Pyramid Match Kernel, Parallel Computing, Cluster of Computers, Support Vector Machine Classifier.*

1. Introduction

An object recognition system considers the problem of recognizing the semantic category of an image. There are many methods that try to increase the recognition rate in this area. In the recent years, new kernel methods that have been introduced are considered as a good approach to improve the support vector machine (SVM) classifier [1] performance. Spatial Pyramid Match Kernel (SPK) [2] is one of these methods that is used to categorize objects in the Caltech101 database.

SPK provides a good recognition rate in this database. Although other methods with a higher accuracy have been introduced, SPK is more applicable, as it is used in many research works.

Most object recognition methods suffer from a heavy computational load. Parallel processing can decrease the running time. Utilizing multi-core CPUs or a cluster of computers is a well-known parallel technique for running different parts of an object recognition algorithm simultaneously. Some parallel recognition methods use this technique.

The MATLAB technical computing language and development environment has been utilized in a variety of fields such as image and signal processing, control systems, modeling, and computational biology [3]. Advances in computer processing power have provided an easy access to multi-processor computers through various methods such as multi-core processors, clusters built from commercial, off-the-shelf components, and a combination of the two. This has made demand of desktop applications similar to MATLAB to find mechanisms for exploiting such architectures. Since many implementations of image processing algorithms are using MATLAB, finding approaches to parallelize them can improve their performance.

This paper presents an approach to parallel computing SPK using MATLAB Parallel Computing Toolbox (PCT) [4]. SPK has 4 steps:

1. SIFT (Scale Invariant Feature Transform) [5] descriptor Extraction.
2. Creating Dictionary by K-Means clustering.
3. Creating Histograms Pyramid of SIFT descriptors.

4. Computing SPK.

Finally, SVM classifier is trained by the pyramid as the feature vector, and SPK as the kernel. These steps impose a heavy computation, so parallel implementation can help improve performance for a better application.

Dependency issues is one of the main problems in parallelization of most algorithms.

Dependencies among tasks have a major impact on running simultaneously different sections of an algorithm. Thus, some solutions should be presented to handle this important challenge. There are different data dependencies between the SPK algorithm steps: K-means requires SIFT descriptors for clustering features; computing histograms requires the result of K-means clustering and the SIFT features.

In this paper, a parallel paradigm of the SPK algorithm is presented. In order to accelerate these steps, the MATLAB Parallel Computing Toolbox and MATLAB MPI are utilized.

2. Related works

So far, SPK has not been implemented by a parallel technique but there exist many pattern recognition algorithms implemented by parallel techniques, and in this section, we will review some of them, particularly K-means, SIFT, and SVM, which are used in this paper.

K-means is a known clustering algorithm in the pattern recognition that is used in our method as well. This algorithm has heavy computing steps; therefore, some parallel techniques have been employed to decrease its time complexity. Our parallel method is based upon a method [6] in which a Java application is implemented to parallelize the K-means algorithm. They distribute a selection of center of clusters among cores. Baydoun et al. have presented an enhanced parallel implementation of K-Means clustering using Cilk Plus and OpenMP on the CPU and CUDA on the GPU. The results obtained are presented for different datasets and images of varying data sizes [7]. Using CUDA and GPU, handwriting recognition is also parallelized [8]. To increase the speed, some part of the recognition method is executed on GPU cores implemented by CUDA platform.

In object recognition algorithms, feature extraction is usually a time-consuming step, so some recent research works have been carried out to parallelize this step. Warn, S et al. have implemented a parallel version of local key and descriptor extraction for SIFT with OpenMP [9] parallelization and GPU (Graphics Processing Unit) execution [10]. Another work in this area is parallelization of SIFT both on a Symmetric Multiprocessor (SMP) platform and a large-scale Chip Multiprocessor (CMP) simulator [11]. A parallel hardware architecture for real-time image feature detection based on the SIFT algorithm has been presented in [12]. It provides the results via a Field-Programmable Gate Array (FPGA). SVM is an algorithm used in SPK. This classifier has some parallel implementations. Distributing, processing, and optimizing the subsets of the training data across multiple participating nodes of the distributed SVM reduce the training time [13].

Tan et al. have proposed a two-level parallel computing framework to accelerate the SVM-based classification by utilizing CUDA and OpenMP [14]. In another work, parallel algorithmic implementations of semi-parametric SVM and Gaussian processes are presented using OpenMP [15].

The MATLAB PCT facilitates parallelization of algorithms such as SPK. Feng H. et al. have suggested the MATLAB PCT to accelerate the SIFT algorithm [16]. Their results show that the parallel versions of the former sequential algorithm with simple modifications achieve the speed-up up to 6.6 times. The parallel MATLAB algorithm of a SVM classifier is implemented as well [17]. This implementation is based upon the message passing interface standard, in which processes coordinate their work and communicate by passing messages among themselves as well as parallel array programming in MATLAB. Using MATLAB Distributed Computing Engine for the plan of parallel genetic algorithm (PGA), a higher speed and a better performance might be obtained [18].

3. Spatial pyramid match kernel

In pattern recognition, defining new Kernels for SVM is an efficient approach. Grauman and Darrell [19] have proposed pyramid match kernel (PMK) using matched local features in feature space. The local features were extracted using Harris detector [20], and their surrounding area was described by SIFT descriptor. Since PMK discards all the spatial information, Lazebnik et al. have introduced an approach, called Spatial Pyramid Match Kernel, adopted from PMK. In the first step, they used a dense regular grid instead of interest points to describe the area via SIFT descriptor. This decision was based upon the Fei-Fei and Perona's evaluation [21], who have shown the dense features work to be better for scene classification.

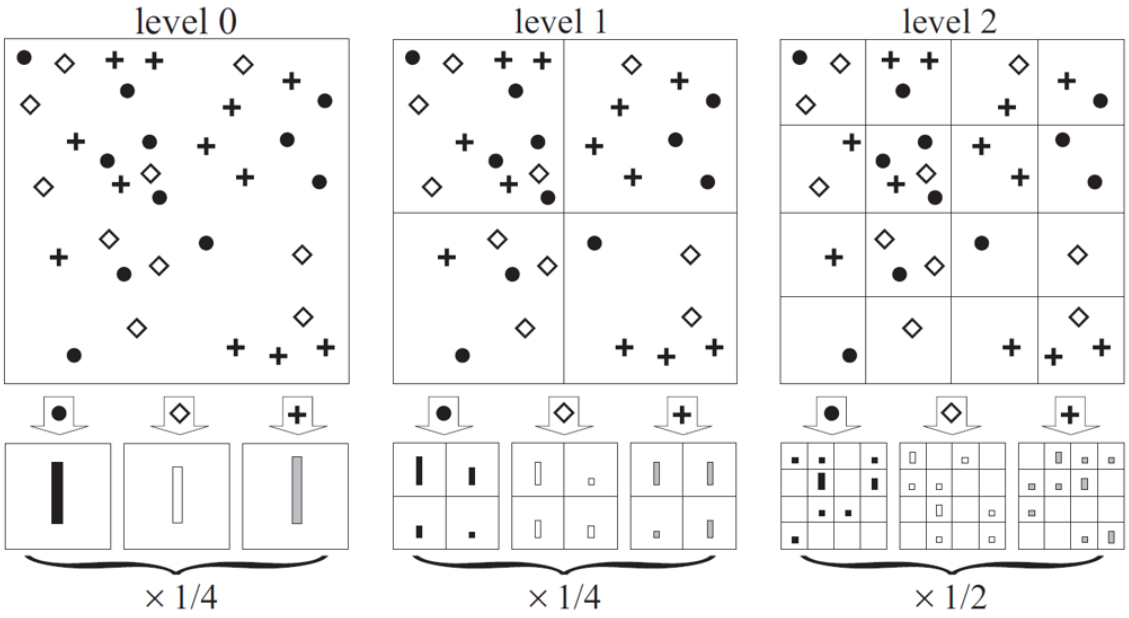


Figure 1. Example of constructing a three-level pyramid. Image has three feature types, indicated by circles, diamonds, and crosses. Each spatial histogram is weighted according to Eq. 2.

In the second step, they performed K-means clustering on SIFT descriptor of patches to form a visual vocabulary. These patches, which had been computed over a grid with a given spacing, have randomly been selected from the training set. In the next step, they computed the pyramid of the histogram by textones of vocabulary. Let X and Y be two sets of vectors in a d-dimensional feature space. Consider a sequence of grids at resolutions 0,…, L such that the grid at level l has 2^l cells along each dimension for a total of $D=2^{dl}$ cells. For each grid in each level, the histograms of features are calculated. Then the concatenation of these histograms forms feature vectors for SVM.

To calculate SPK, the matches among images are computed. Note that matches found at higher levels are more valuable, so higher weights are assigned to them compared with matches at lower levels. The number of matches at level l is given by the histogram intersection function according to (1).

$$I(H_X^1, H_Y^1) = \sum_{i=1}^{D} \min(H_X^1(i), H_Y^1(i)) \quad (1)$$

H_X^1 and H_Y^1 denote the histograms X and Y at this resolution, so that $H_X^1(i)$ and $H_Y^1(i)$ are the numbers of points from X and Y that fall into the cell of the grid. In the following, we will abbreviate $H_X^1(i), H_Y^1(i)$ to I^l. The number of matches found at level l also includes all matches found at the finer level l+1. The number of new matches found at level l is given by $I^l + I^{l+1}$ for l=0,…, L-1. The weight associated with level l is set to $\frac{1}{2^{L-l}}$ that is inversely proportional to the cell width at that level. The pyramid match kernel is defined as (2).

$$k^l(X,Y) = I^L + \sum_{i=0}^{L-1} \frac{1}{L-1}(I^l + I^{l+1}) = \frac{1}{2^L} I^0 + \sum_{i=1}^{L} \frac{1}{L-1+1} I^l \quad (2)$$

The defined pyramid match kernel is the value of matches for one type of features. The paper quantizes all feature vectors into M discrete types using K-means clustering algorithm, and adopts the simplifying assumption that only features the same type that can be matched to one another. Then the definition of spatial pyramid match kernel using the sum of separate channel kernels based on (3) are performed. Each channel m gives us two sets of 2D vectors, X_m and Y_m, representing the coordinates related to features of type m found in the respective images [2].

$$K^l(X,Y) = \sum_{m=1}^{M} k^L(X_m, Y_m) \quad (3)$$

M is the number of feature types (clusters) that are obtained using K-means clustering to cluster all features. In another word, M is equal to K in K-means clustering. Figure 1 shows a three-level pyramid. The Kernel function is calculated for every two image in the training set.

4. MATLAB parallel computing toolbox

Parallel Computing Toolbox (PCT) solves computational and data-intensive problems using multi-core processors, GPUs, and computer clusters [3]. The toolbox provides 12 workers (MATLAB computational engines) to execute applications locally on a multi-core desktop or more workers on clusters using MATLAB Distributed Computing Server. The algorithms can be parallelized in MATLAB without additional coding for specific hardware and network architectures. PCT allows us to assign each part of an algorithm to each worker as a job. This is done in the client MATLAB session. Workers run their jobs simultaneously, so the algorithm runtime is reduced. Here, some capabilities of PCT for parallel programming are mentioned:

1. Parallel for-Loops (parfor): Many applications involve multiple segments of code, some of which are repetitive. Often for-loops might be employed to solve these cases. The ability to execute code in parallel, on one computer or on a cluster of computers, can significantly improve the performance in many cases.
2. Distribution of data: If the code has an array that is too large for the computer memory, it cannot be easily handled in a single MATLAB session. The PCT software allows distributing that array among multiple MATLAB workers, so that each worker contains only a part of the array. Each worker operates only on its own part of the array, and workers automatically transfer data among themselves when necessary. The SPMD (Single Program Multiple Data) statement lets us run a part of code on multiple data simultaneously.
3. Definition of jobs and tasks: When working interactively in a MATLAB session, work can be off-loaded to a MATLAB worker session to run as a job. The command to perform this job is asynchronous, which means that the client MATLAB session is not blocked, and the interactive session can be continued while the MATLAB worker is busy evaluating the code. This can be done through defining distributed parallel jobs. If the application involves large datasets on which simultaneous calculations are performed, a parallel job with distributed arrays can be used. If the application involves looped or repetitive calculations that can be performed independently, a distributed job might be appropriate. Each job can have multiple tasks, so that every task is run on a separate worker.

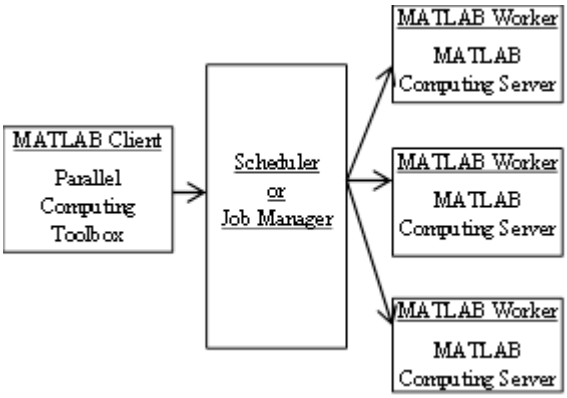


Figure 2. Basic PCT.

Management of the whole Parallelization, as shown in figure 2, is the duty of the MATLAB job manager or third-party scheduler.

5. Parallel implementation of spatial pyramid match kernel

The first aspect of the implementation is to run different independent parts of the algorithm simultaneously. Thus, the algorithm steps are distributed among workers. The second aspect includes parallelizing special steps, i.e. K-means and SPK computing, on the cluster. It means that some sub-parts have the possibility of parallelization in addition to distributing the main parts of algorithm on workers.

In the sequential algorithm, SIFT descriptors extraction from the training set and K-means clustering run sequentially but these two steps are independent from each other. Forming vocabulary can be done using SIFT descriptors of some random images by the K-means algorithm, so it is independent from extracting SIFTs from all the images. Thus, two parallel jobs on separate job managers are defined. Job 1 selects some images randomly, and then runs K-means clustering on the features of these images to form the vocabulary. Job 2 is in charge of extracting SIFT descriptors for all images and running other steps of the algorithm including computing histograms as well as their pyramid and constructing the kernel matrix. The different parts of the first step (job 1) are dependent on each other. For example, the SIFT descriptors of some random images should be calculated, and then these features are clustered using K-means. Hence, they cannot run simultaneously. However, the sub-parts of each step can run in parallel. Extracting SIFT

descriptors can be distributed on computers. Also the K-means algorithm can be executed in parallel.
Different parts of Job 2 are dependent on each other as well. For example, at first, SIFT descriptors should be extracted, then pyramid of histograms calculated. However, like the previous Job, these parts can be run in parallel.

5.1. Definition of Job 1

As mentioned earlier, this parallel job of PCT has just one task that is executed on several workers simultaneously. At the beginning, the job manager distributes random-selected images among workers, which run this job (capabilities 1 and 3 of PCT, mentioned before). Each worker describes patches of its part of the images. After extracting SIFT descriptors, K-means clustering should be performed. K-means has two steps:
1. Assignment step: Assign each data to the cluster with the closest mean.
2. Update step: Calculate the new means to be the centroid of the data in the cluster.
The first step is the time-consuming part because distances between all vectors and centroids must be calculated. Thus, this step is tried to be parallelized via distributing calculation of the distances. Therefore, each worker is responsible for assigning its part of the SIFT descriptors to the nearest cluster.
The second step requires all the data that is distributed among workers. In the first iteration, centroids are selected randomly, so there are two solutions for determining them:
1. Transferring all extracted SIFT descriptors from workers to a specified worker, then selecting centroids by that worker, and finally, broadcasting them to other workers.
2. Selecting some random centroids by each worker from its part of the extracted SIFT descriptors, then job manager concatenates them by MPI concatenate function to create an array of centroids and broadcast them to all workers so that all workers have the same centers.
There is a communication overhead in the first solution because intense data must be transferred, whereas transmitted data and centroids at the second solution are less. The second solution is used in the experiments. After selecting the first centroids, each worker assigns data to clusters. For the next iterations, each worker cannot update centroids alone. Some data should be transferred among workers to calculate new centroids. The following data is gathered from all workers using MPI functions by the job manager to update centroids:

1. Sum of data distances for each cluster
2. Number of data belonging to each centroid

Job manager broadcasts this information to all workers. New centers are computed based on new posteriors using this information; therefore, workers update their centroids. These two steps repeat until the termination condition is met. At the end of the algorithm, the last centroids should be saved for later steps of SPMK; thus, a specified worker, for example, worker number one saves centers in a file. All of these steps are shown in figure 3**Error! Reference source not found.** Pseudo-code of Job 1 is also represented in figure 4. The code represented in figure 4 is executed on all of the workers in parallel. The parts that are relevant to gathering or distributing information are automatically performed using job manager. These parts include:

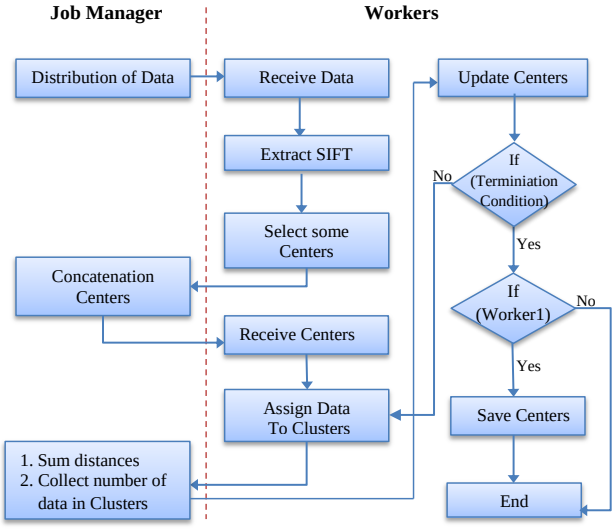


Figure 3. Steps of job 1.

```
Function Paralleljob1(Random-images)//Run this function as parallel
job on all of workers
Begin
      Local_part_image=distribute(Random_images)
      for Each image of local_part_image do
            Call Generate SiftDescriptor WITH image RETURNING
SiftDescriptor
      end
      N-Clusters=200
      Iterations=100
      CenterLab=N-Clusters/NumLabs of Random selected SiftDescriptors
      Centers=Concatenate(CenterLab) //using gcat MPI function of
MATLAB
      for Number of K-means Iterations do
            Assign each SiftDescriptor to nearest center
            for each center of N-Clusters do
                  plus number of points in cluster //using gplus MPI
            function
                  Plus Euclidean distance between SiftDescriptors and
                  center //using gplus MPI function of MATLAB
            end
            Upfate Centers
            if Labindex=1 then
                  Save Centers
            end
      end
end
```

Figure 4. Pseudo-code of Job 1 that runs on workers simultaneously. Some functions not interfering with parallel implementation are considered as black box and are just called.

Local_part_image = Distribute (Random-images): job manager distributes random images on workers to extract SIFT descriptor.

Plus, number of points in cluster: Job manager gathers the number of images from each cluster and broadcasts them to all of the workers for updating centroids.

Plus, Euclidean distance between SiftDescriptors and center: job manager gathers Euclidean distances between sift descriptors and centers then broadcasts them to all of the workers for updating centroids.

The remainder of the code is executed in parallel on all of the workers. Also, these parallel parts are represented in the worker section of Figure 3.

5.2. Definition of Job 2

Job 2 performs the following steps:

1. Extracting SIFT descriptors for training set
2. Creating Histograms Pyramid of SIFT descriptors
3. Computing Spatial Pyramid Match Kernel

Job manager distributes images among workers, and every worker extracts SIFT descriptors of its part of the images. Also, workers compute histograms of these features and construct a pyramid of histograms. Until now, workers do not require to communicate with each other but they must transfer data between themselves for computing spatial pyramid match kernel. Kernel is a symmetric matrix, where element $K_{i,j}$ indicates the value of kernel function corresponding to images, j. Each worker alone cannot compute the kernel value corresponding to its own images because they need histograms of other images to calculate matches between them. Here, two solutions are suggested for the workers required communication for transferring histograms of their own images:

1. Using the MPI function to concatenate all histograms from workers, and then broadcasting them by the job manager. Each worker calculates the rows of its part of the kernel matrix (concatenation method).
2. Computing half of the kernel matrix by transmitting half of the data (round method). The designed procedure is depicted in Figure 5.

A loop was defined, whose number of iterations is half of the number of workers. At the first iteration, every worker sends its data to the next one, and at the same time receives data from the former worker. At the next iterations, the labindex-th worker (worker[labindex]) sends histograms to worker [labindex+2] and receives histograms from worker [labindex-2], and so on.

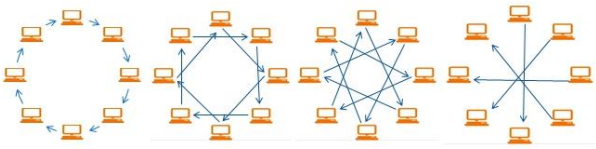


Figure 5. Round method for communication between 8 workers: each worker sends data to another one and receives data from others simultaneously in every iteration.

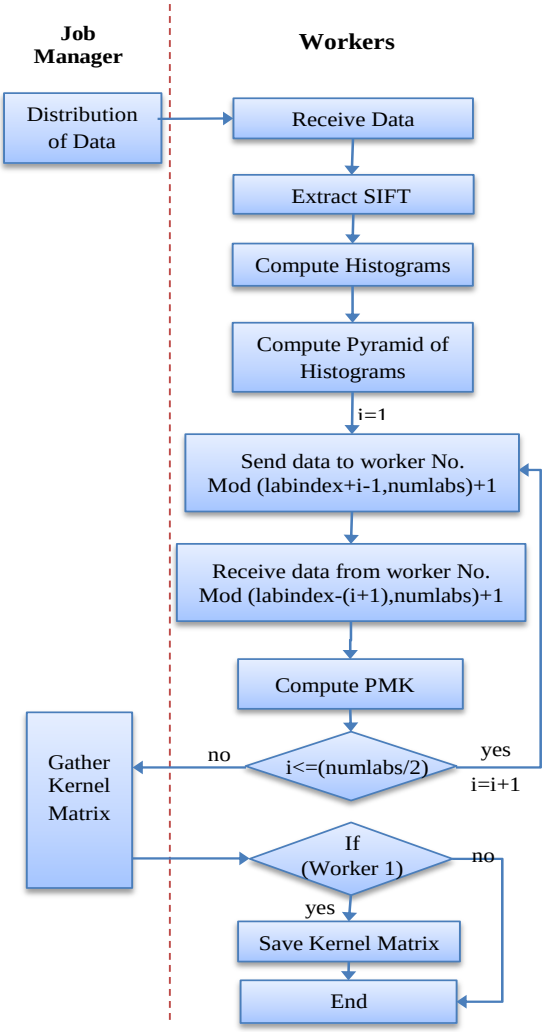


Figure 6. Steps of Job 2

At the end of the iterations, workers compute rows of the kernel matrix corresponding to their part of images and receive images according to (3). Thus, this method helps workers avoid waiting for a long time to communicate. Figure 6 shows the configuration of this job. The pseudo-code of parallel job 2 is illustrated in figure 7.

The code represented in figure 6 is executed on all of the workers in parallel. The parts that are relevant to gathering or distributing information are automatically performed using the job manager. These parts include:

Local part image=distribute(images): The job manager distributes random images on workers to extract SIFT descriptors.

Gather LocalKernelMatrix from all workers to build KernelMatrix: job manager should collect information from all workers to build kernel matrix.

KernelMatrix: job manager should collect information from all workers to build kernel matrix.

The remainder of the code required to run on all the workers in parallel, as presented in the worker part of figure 6.

At the end of the procedure, the MPI function gathers different parts of kernel matrix that are in workers' memory, and then they are saved by one of the workers, for example, worker number one.

5.3. Time complexity of algorithm

To obtain the time complexity of the sequential algorithm, 4 parts might be considered:

1. Computing SIFT descriptors: $O(n*g)$, n: number of images, g: number of grids.
2. K-means algorithm: $O(mkl)$, m: number of random images used in clustering, k: number of clusters, l: number of iterations.
3. Computing histograms and their pyramid: $o(n*2^p)$, p: number of patches.
4. Computing Kernel Matrix: $O(n^2)$.

Overall, the time complexity of the sequential algorithm is $O(n^2)$.

In the parallel algorithm, all parts are based on distributing data, thus the time complexity of parts is as follows:

1. $O(n*g/N)$: Without any communication time, N: number of cores (workers).
2. $O(mkl/N)$: The time complexity for computing $+O(k)$: The time complexity of communication for transferring centers of clusters. k is always much smaller than m, so the whole complexity of this part is $O(mkl/N)$.
3. $O(n*2^p/\mathrm{N})$.
4. $O(n^2/N)$: The time complexity for computing $+O(\mathrm{N}/2*\mathrm{n}/\mathrm{N})$: The time

complexity of communication for transferring histograms.

5. The whole complexity of this part is $O(n^2 / N)$.

Overall, the time complexity of the parallel algorithm is $O(n^2 / N)$. The interesting point is that the time complexity of communication in the problem with large datasets has a very smaller growth compared to the computation time. Thus the parallel algorithm can be effective and optimum.

Experiments

For our experimental evaluation, the Caltech101 database was exploited [22]. Caltech is an object image database utilized for object recognition method assessment. Pictures of objects belong to 101 categories. There are 40 to 800 images per category. Most categories have about 50 images. They have been collected in September 2003 by Fei-Fei Li et al. The size of each image is roughly 300x200 pixels. The SVM classifier is trained on SIFT features from 200 random images. The kernel of SVM is spatial pyramid match kernel calculated for a pair of images in the parallel section. The trained model of SVM can be used to test unseen images in the test phase. Due to focuses on the train phase, the detail of test is not considered in this paper but for evaluation of the results compared to serial version of algorithm, 100 images are randomly selected to test the model.

There are four workers defined on each computer, one for every core, i.e. a total of 20 workers. Hence, worker in this paper means core. One of the computers is designated as the job manager.

Some parameters that are used in the experiments are as follow:

1. Number of histogram levels: 4
2. Number of K-means clusters: 200
3. Number of pixels of image grid: 8

Job 1 and job 2 are evaluated on different numbers of images and workers for several times to compute how much speed-up is achieved compared to the serial algorithm. In parallel computing, speed-up refers to how faster a parallel algorithm is compared to the corresponding sequential algorithm. Speed-up is the ratio of the sequential algorithm execution time to parallel algorithm. Linear speed-up or ideal speed-up is obtained when speed-up is equal to the number of processors.

```
Function Paralleljob2(images)//Run this function as parallel job on
all of workers
Begin
      Load centers of clusters
      Local_part_image=distribute(images)
      for Each image of Local_part_image do
            Call GenerateSiftDescriptor WITH image RETURNING
            SiftDescriptor
            Call BuildHistogarm WITH SiftDescriptor,Centers ReTURNING
            Histogarm
            Call BuildPyramid WITH Histogram RETURNING Pyramid
      end
      for i=1 to (NumLabs/2)-1 do
            LabTo =mod(labindex+i-1, numlabs)+1
            LabFrom = mod(labindex - (i+1), numlabs)+1
            Received_pyramid=labSendReceive(lab To, LabFrom,Pyrsmid)
            //a function from MATLAB PCT
            Call BuildKernel WITH Recienvedpyramid, Pyramid RETURNING
            LocalKerenMatrix
      end
      Gather LocalKernelMatrix from all workers To build KernelMatrix
      //using gather function of MATLAB PCT
      if Labindex=1 then
            Save KernelMatrix
      end
end
```

Figure 7. Pseudo-code of Job 2 that runs on workers simultaneously. Some functions that do not interfere with parallel implementation are considered as black box and are just called.

6.1. Evaluation of job 1

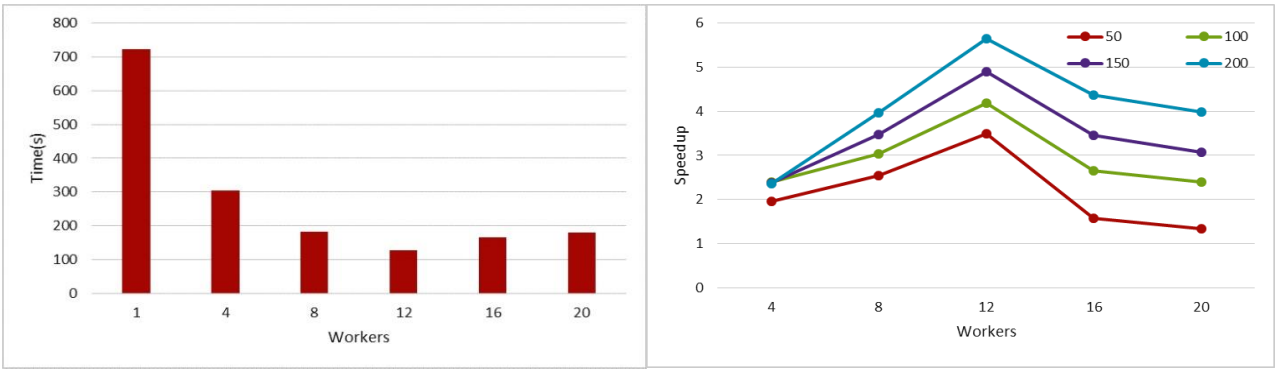
Job 1 extracts SIFT descriptors for randomly selected images, and then executes K-means. Termination of K-means occurs when the clustering error is less than 0.1 or the number of iterations exceeds 100. Speed-up depends on the number of workers but it does not increase linearly. 12 workers give us the highest speed-up. Figure 8(a) shows changes in job runtime with different numbers of workers. The decrease in the runtime with increase in the number of workers is not true for all cases.

The reason for this inconsistency is that when the number of workers increases, CPUs spend more time to send, receive or wait for the data, so computation ratio decreases. Therefore, there is a trade-off between the communication cost and speed-up. Figure 8(b) illustrates the speed-up ratio of job 1 on different numbers of images. Notice that each image has roughly 1000 extracted features. Thus for 200 selected images, K-means clustering is executed on 200000 data points. As the number of data points goes up, the speed-up ratio of the parallel process increases. More data points cause computation to take more time than communication because the number of centers is constant. Figure 9(a) shows the ratio of computation time to total time. As the number of workers is increased, this ratio decreases. Also, the total amount of data that is transferred among workers is shown in this diagram. The relationship between increasing data transformation and number of workers is linear.

Figure 9(b) represents the communication time and the ratio of communication cost related to total time of running job 1. As it is shown, increasing the number of workers causes increase in the communication cost, for example, 20% of time is taken for communication when the number of workers is 12. Although this is a lot of time of running, analyzing two jobs totally shows a good speed-up.

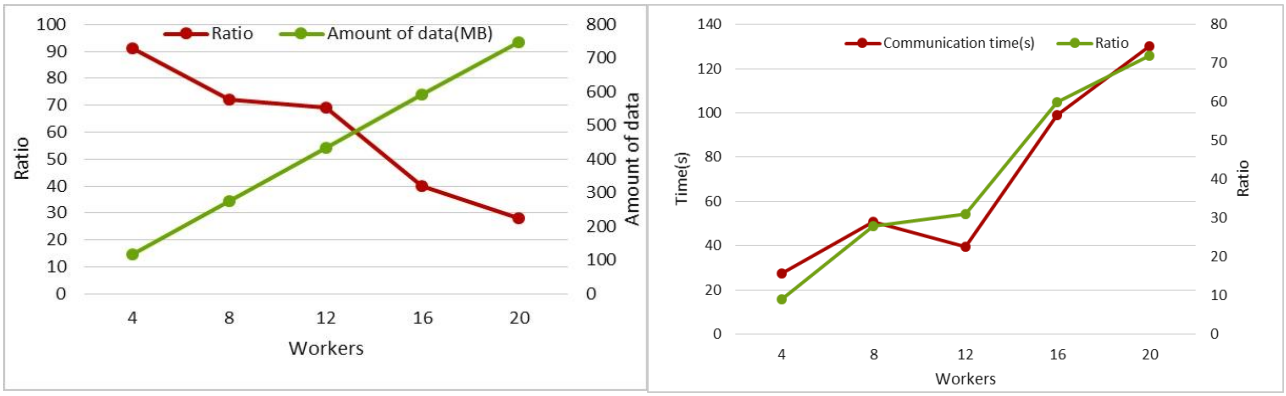
6.2. Evaluation of job 2

Job 2 extracts SIFT descriptors of all images, and then computes pyramid of histogram and SPK using clusters obtained by job 1. Computers access map network drive to read the necessary data. For the training stage, 15 or 30 images are selected from each category.



(a) (b)

Figure 8. Time taken to run Job 1 on 200 images with different numbers of cores (workers) (a). Speed-up with different number of images (50, 100, 150, 200) on Job 1 (b).



(a) (b)

Figure 9. Transferred data in Mb and ratio of computation to total time in Job 1 (a). Communication time and ratio of communication to total time in Job 1 (b).

Different speed-ups result with varying numbers of workers. Evaluation is done in two methods of data transmission:

1.1. Round method

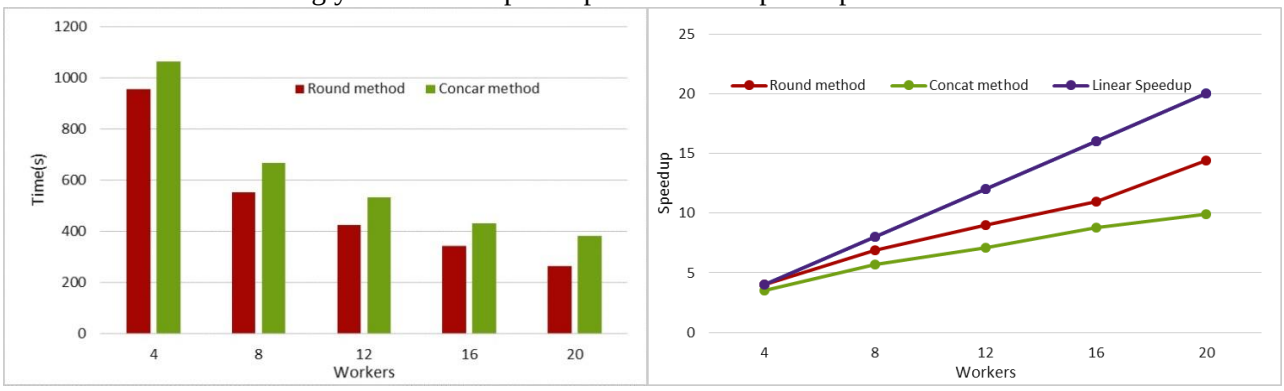
2.1. Concatenation method.

Figure 10(a and b) show the obtained job 2 speed-up and runtime. The number of images per category is 30. As the number of workers is increased, the runtime decreases; however, this time reduction (speed-up) is not linear because of the communication time but the acceptable speed-up is achieved. The maximum speed-up (15) is observed using the round method with 20 workers. An important point about speed-up enhancement is acceleration. Increasing workers enhances speed-up but the diagram deviates from a linear speed-up. Thus, a trade-off between cost and speed-up in parallelization is necessary. The best choice for the number of workers depends on the application of the method. In some situations, the algorithm runtime is the most important issue. Hence, obtaining a maximum speed-up using full resources is a rational decision.

Figure 11(a) demonstrates the ratio of computation to total time and transformation data of job 2 in details for different numbers of workers. Although the total time is reduced for a higher number of workers, the communication time and data transformation between all workers increase and the computation time decreases. In this situation, a worker should send data to more workers. Figure 11(b) illustrates the ratio of communication time to the total time. It is about 50% for 20 workers. It means that the communication cost strongly affects the speed-up improvement. At the end of this job, one of the workers trains SVM using SPK matrix obtained by job 2 and pyramid of histograms as feature vectors. All of these parallel steps can be repeated for testing a remarkable number of images; otherwise, with a few number of images, running the serial algorithm takes less time. Thus parallelization is not beneficial. About the efficiency of implementation, it must be emphasized that parallelization does not affect the accuracy of the method to recognize category of objects, thus parallel computing saves time and offers a better performance.

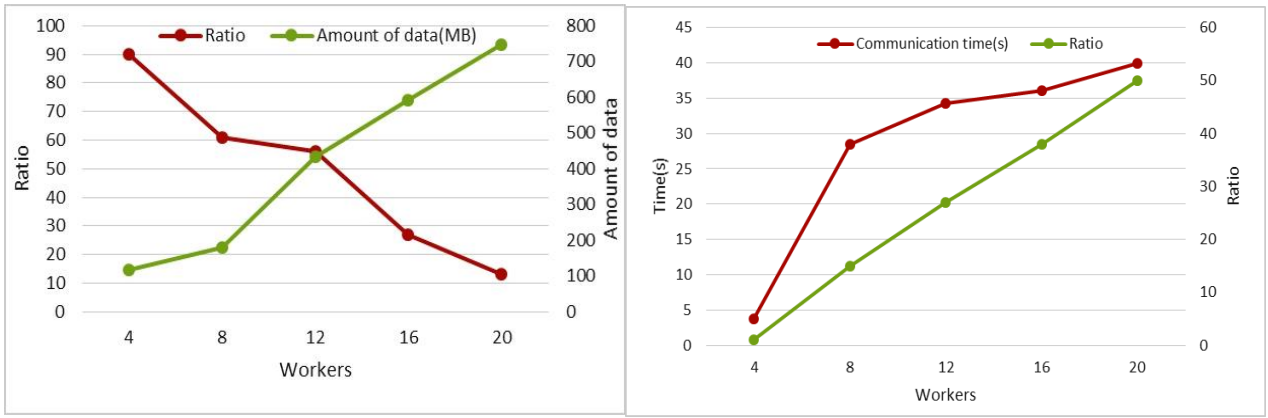
6.3. Merging of two jobs

In the previous section, the best performance of each job was discussed independently. Job 1 achieves the best speed-up with 12 workers, and job-2 with 20 workers. Resources have to be shared between these two jobs to run simultaneously. Hence, it is important to divide 20 workers in such a way that the best performance is obtained. It is obvious that job 2 needs data from job 1 for making a histogram; thus the best performance can be achieved when job 1 and the extracting SIFT in job 2 finish simultaneously. Job 1 and just extracting SIFT in job 2 can be performed in parallel. The best situation is achieved when these two parts end simultaneously, and then data from job 1 is employed by job 2 to continue the work. Studying the experiments, if 2 workers are used for job 1 and 18 workers for job 2, job 2 does not have to be idle for job 1, and in this situation, 13 is the best speed-up.



(a) (b)

Figure 10. Time taken to run job 2 in two methods, round and concatenation (a). Speed-up of two methods, round and concatenation.



(a) (b)

Figure 11. Transferred data in Mb and ratio of computation to total time in Job 2 (a). Communication time and ratio of communication to total time in Job 2 (b).

7. Conclusion

This paper has presented a parallel version of an object recognition method. This method applies a spatial pyramid match kernel to a support vector machine classifier to recognize image categories. Our implementation parallelizes different steps of calculating kernel on a cluster of computers. The MATLAB parallel computing toolbox was used to distribute jobs on computers. Parallel algorithm is thirteen times faster than the serial version for a configuration up to 5 Quad processors. The results obtained show that the accuracy of the method is not affected by parallelization. Thus the implementation is well-suited for applying in other research works that use this kernel.

8. References

[1] Vapnik, V. N. & Kotz, S. (1982). Estimation of dependences based on empirical data. Springer-Verlag New York.

[2] Lazebnik , S., Schmid C. & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, New York, USA, 2006.

[3] Sharma, G. & Martin, J. MATLAB®: a language for parallel computing, (2009). International Journal of Parallel Programming, vol. 37, pp. 3-36.

[4] Halkidi, M., Batistakis, Y. & Vazirgiannis, M. (2001). On clustering validation techniques, Journal of intelligent information systems, vol. 17, pp. 107-145.

[5] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints, International journal of computer vision, vol. 60, pp. 91-110.

[6] Ramesh, V., Ramar, K. & S. Babu, (2013). Parallel K-Means Algorithm on Agricultural Databases. International Journal of Computer Science Issues (IJCSI), vol. 10, pp. 46-56.

[7] Baydoun, M., Dawi, M. & Ghaziri , H. (2016). Enhanced parallel implementation of the K-Means clustering algorithm. 3rd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), Lebanon, 2016.

[8] Askari, M., Asadi, M., Asilian Bidgoli, A. & Ebrahimpour, H. (2016). Isolated Persian/Arabic handwriting characters: Derivative projection profile features, implemented on GPUs. Journal of AI and Data Mining, vol. 4, pp. 9-17.

[9] Dagum, L. & Menon, R. (1998). OpenMP: an industry standard API for shared-memory programming, Computational Science & Engineering, vol. 5, pp. 46-55.

[10] Warn, S., Emeneker, W., Cothren, J. & Apon A. W. (2009). Accelerating SIFT on parallel architectures, IEEE International Conference on Cluster Computing and workshops, Louisiana, USA, 2009.

[11] Feng, H., Li, E., Chen, Y. & Zhang, Y. (2008). Parallelization and characterization of SIFT on multi-core systems, IEEE International Symposium on Workload Characterization, Seattle, USA, 2008.

[12] Peng, J., Liu, Y., Lyu, C., Li, Y., Zhou, W., & Fan, K. (2016). FPGA-based parallel hardware architecture for SIFT algorithm, IEEE International Conference on Real-time Computing and Robotics (RCAR), Angkor Wat, Cambodia , 2016.

[13] Caruana, G., Li, M. & Qi, M. (2011). A MapReduce based parallel SVM for large scale spam filtering, Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Shanghai, China, 2011.

[14] Tan, K., Zhang, J., Du Q. & Wang, X. (2015). GPU parallel implementation of support vector machines for hyperspectral image classification, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 8, pp. 4647-4656.

[15] Díaz-Morales, R. & Navia-Vázquez, Á. (2016). Efficient parallel implementation of kernel methods, Neurocomputing, vol. 191, pp. 175-186.

[16] Cao, H. & Chen, J. (2012). Multicore Computing for SIFT Algorithm in MATLAB® Parallel Environment, 18th International Conference on Parallel and Distributed Systems, Shenzhen, China, 2012.

[17] Guilfoos, B., Gardiner, J., Chaves, J. C. Nehrbass, J., Ahalt, S. & Krishnamurthy, A. (2006). Applications in Parallel MATLAB, HPCMP Users Group Conference, DC, USA, 2006.

[18] Guifen, C., Baocheng, W. & Helong, Y. (2007) . The implementation of parallel genetic algorithm based on MATLAB, Advanced Parallel Processing Technologies, stockholm, Sweden, 2007.

[19] Grauman, K. & Darrell, T. (2005). The pyramid match kernel: Discriminative classification with sets of image features, Tenth IEEE International Conference on Computer Vision, Beijing, China, 2005.

[20] Harrism, C. & Stephens, M. (1988). A combined corner and edge detector, Alvey vision conference, Manchester , England, 1988.

[21] Fei-Fei, L. & Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories, IEEE Computer Society Conference on Computer Vision and Pattern Recognitio, DC, USA, 2005.

[22] Fei-Fei, L., Fergus, R. & Perona, P. (2007). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories, Computer Vision and Image Understanding, vol. 106, pp. 59-70.

نشریه هوش مصنوعی و داده کاوی

موازی سازی الگوریتم هسته تطبیقی هرمی مکانی با استفاده از کلاستری از کامپیوترها برای شناسایی اشیا در تصاویر دیجیتال

اعظم اصیلیان بیدگلی[1]، حسین ابراهیم پور کومله[2]، میثم عسکری[2] و سیدجلال الدین موسوی راد[2]

[1] دانشکده مهندسی کامپیوتر، دانشگاه کاشان، کاشان، ایران - دانشکده مهندسی برق و کامپیوتر، موسسه آموزش عالی پویش، قم، ایران.

[2] دانشکده مهندسی کامپیوتر، دانشگاه کاشان، کاشان، ایران.

ارسال ۲۰۱۶/۰۴/۲۵؛ بازنگری۲۰۱۷/۰۶/۱۲؛ پذیرش ۲۰۱۷/۰۷/۱۹

چکیده:

این مقاله پیاده‌سازی موازی از الگوریتم هسته تطبیق هرمی مکانی را ارائه می‌دهد. در سال‌های اخیر، الگوریتم هسته تطبیق هرمی مکانی یکی از هسته‌های پرکاربرد برای طبقه‌بند ماشین بردار پشتیبان در شناسایی اشیا با دقت بالا محسوب می‌شود. در این پیاده‌سازی توابع واسط تبادل پیام متلب با استفاده از دو مدل موازی سازی داده و موازی سازی کار به بهبود کارایی طبقه‌بندی تصاویر کمک شایان توجهی کرده‌اند. الگوریتم هسته تطبیق هرمی مکانی موازی روی کلاستری از کامپیوترها اجرا شده و باعث کاهش زمان اجرای الگوریتم نسبت به الگوریتم ترتیبی می‌شود. میزان تسریع بدست آمده الگوریتم موازی با توجه به تعداد پردازنده و تعداد هسته‌های آن می‌تواند متفاوت باشد. در این پیاده‌سازی میزان تسریعی برابر ۱۳ با استفاده از ۵ کامپیوتر با پردازنده های ۴ هسته‌ای بدست آمده است.

کلمات کلیدی: شناسایی الگو، هسته تطبیق هرمی مکانی، محاسبات موازی، خوشه‌ای از کامپیوترها، طبقه‌بند ماشین بردار پشتیبان.