Optimization of Support Vector Machine on Multi-core processor with OpenMP

Sumit Patel, Bhadreshsinh Gohil, Dr. M. B. Potdar

Abstract— In this paper, multicore processors and OpenMP are used for optimize the SVM. The SVM is machine learning algorithms, which is widely used in image classification. There are different methods for clustering data for SVM training phase. Here, we used KMeans clustering, because it gives better result than others methods. We used different size of data for analyse the performance of KMeans and SVM. We got more speedup for large size data on multicore processors. Here, we used Open Multi Processing library for parallelizing OpenCV (Open Computer Vision) programs, which is library for image processing. Here, we discussed the performance of OpenCV programs with OpenMP for small size data and big size data.

Index Terms— KMeans Clustering, Multicore, OpenCV, OpenMP, Parallelization, Speedup, SVM, Threads.

1 Introduction

THERE are many image processing tools and algorithms. It includes KMeans clustering, K-Nearest, Haar Classifier, Support Vector Machine [1] and Wavelet transformation, etc. This tools and algorithms are used in various applications and in various areas [2]. Here, we used SVM for image classification because of its functionality and accuracy. There are various versions of SVM. We used OpenCV SVM source, in which libsvm are used by OpenCV. We used OpenCV library for image processing operations. OpenCV provides different kernels like linear kernel, polynomial kernel and RBF kernel for SVM [3]. We tested different kernels for different types of data. We used KMeans for making clusters of data, which were used in SVM training phase for assigning labels to training data. Yukai Yao, Yongqing Yu, Yang Liu and Weiming Lv suggest that kmeans can work on small data and reduce the training and prediction time of SVM [4]. In past, there was single core or dual core processors for processing and other computation tasks. Nowadays companies make quad core, octa core, and many core processors like Intel Xeon E5-2650. With multi core processors we can get better speedup for various programs. Also it is applicable to image processing algorithms. For making parallel programs we can use CUDA, OpenMP, Intel TBB and Message Passing Interface. OpenMP provides parallelization for shared memory architecture [5], [6].

2 Background

Kun Tan et al. use Support Vector Machine as classifiers for hyperspectral image classification. For making it faster they proposed two-level parallel computing frameworks. They used CUDA for GPU based optimization and OpenMP for task-level parallelism. Because of more than 200 channels in hyperspectral images, data analysis become complex and as data size is increased, time for computation with complexity also is increased. In that paper, NVidia GPU and parallel library CUDA and OpenMP are used with SVM. SVM with GPU take less time than libsvm for training data sets. So with parallel library and GPU classification of images can be done in less time [7]. In image segmentation, partitioning of images are performed. Different parts of original image have different objects and characteristics. For getting information about them, transformation and rotation are required. For that various algorithms are used, but traditional algorithms are time consuming for large size images. With advancement in multicore architecure, methods of programming are changed. OpenMP provides parallelism explicitly for multicore processors. Authors proposed an experimental method for image segmentation with parallel wavelet transform. They used OpenMP parallel library for making parallel faster execution [8]. Rashmi C et al. presented various parts detection of face using Haar Classifier. They used shared memory concept of OpenMP for parallel programming. Detection of front face, eye, nose, eye brow, mouth, hand and so forth require more time. The task of face part detection is not dependent on each other. So OpenMP is useful here, because it provides task-level parallelism and they got different speedup for different types of faces [9]. In this paper [10], there are two phases in SVM classification. For training phase, sequential minimal optimization (SMO) is widely used. To avoid iterative complexity, analytical approach is used for solving sub quadratic programming. It reduces storage space because in this approach, storing of kernel matrix is not required. As per authors view, to accelerate training of large data sets by SMO, there are two ways. First is that with optimization of working data sets, reduce the number of iterations. And second is use parallel version of SMO, which use multithreading approach to reduce the running time. They used OpenMP for parallelization. With OpenMP they got more than 150% speedup. Suli Zhang et Al. [11] said that because of powerful camera and new technologies images are become larger and their size increasing day by day. Their parallel dilate algorithm for signal processing be-

- *Sumit Patel is currently pursuing masters degree program in computer engineering in Gujarat Technological University PG School, Ahmedabad, India, E-mail: smpatel1992@gmail.com*
- *Bhadreshsinh Gohil is currently assistant professor in Gujarat Technological University PG School, Ahmedabad, India, E-mail: bhadresh.wimc@gtu.edu.in*
- *Dr. M.B.Potdar is currently project director in Bhaskaracharya Institute of Space Application and Geo-informatics, Gandhinagar, Gujarat,India, E-mail:mbpotdar11@gmail.com*

International Journal of Scientific & Engineering Research Volume 7, Issue 5, May-2016
ISSN 2229-5518

came slower because of bigger size. So they used MP and MPI for parallelizing algorithm. And their new algorithm was faster than old one. Authors said that we can get parallel computing with distributed architecture system and parallel system [12]. It depends on behaviour of application and format of processing data. Lan Xiaowen said that OpenMP is useful only for multicore processors. Because in OpenMP, program compile with inbuilt directives (#pragma), which cannot give advantage of OpenMP on single core processors [13]. Greg Slabaugh et al. used OpenMP for image processing applications like image warping, binary morphology erosion & binary morphology dilation, median filtering and normalization on multicore processors [5]. For detecting face in image and video P. E. Hadjidoukas et al. use OpenMP in their system. They used different images for analysed the result of new system with OpenMP. Their algorithm was based on neural networks. In sequential algorithm, they could process only 11 images per sec on quad core system. And their parallel version algorithm provided processing more than 25 images per second. For that they used three or more threads [14]. Liuyang Fang et al. presented the paper on ZSMS images processing with parallel libraries. Their implementation was based on MOC. They used CUDA, OpenMP and distributed memory parallel library MPI. They used CPUs and GPUs both and their system took 86.10 seconds execution time for 12 images [15]. Sumit Patel et al. said that CUDA library on Nvidia cards gives higher performance than OpenMP and MPI. But OpenMP is easier than others for parallelizing the sequential algorithms [18].

3 Proposed System

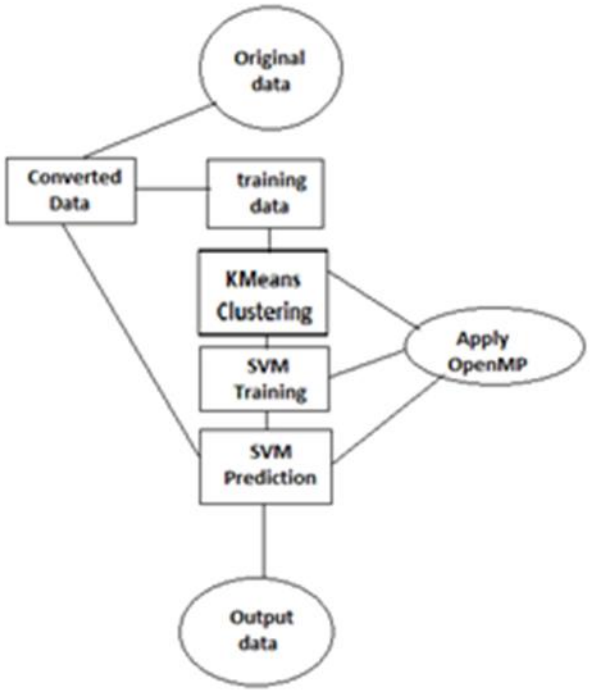

Fig. 1. SVM with OpenMP

As shown in Fig. 1, in our system, first we converted data in suitable format and select 30 % of total data as training data. Then we applied KMeans Clustering before SVM training phase. It divides data into different clusters. After training step, we applied SVM on whole data for prediction. In Output data, we got classified image as output of program. And for parallelization we applied OpenMP on KMeans Clustering, SVM training step and SVM prediction step.

4 Methodology

To perform image processing operation like classification on data, we choose SVM method. We used OpenCV library for implement SVM on different processor. There are different versions of SVM. OpenCV uses libsvm for SVM functions like svm training and svm predictions. Firstly we selected the data files in text format for loading data in program. Then we made one image matrix on of their files. Then we divided the data in training data and testing data in 30:70 ratios. We used 512 by 512 size data and 3000 by 3000 size data. After that we used KMeans clustering for dividing data in clusters. And they are used for assign labels. Generate labels is possible with different techniques. Randomly generation of labels for classification in svm is also used. It is fast in some points, but its result is worse than KMeans clustering. Also parallelization for random number is tough and accuracy of svm also is decreased. Use K-Means for creating cluster in SVM is better than other algorithms. It gives better performance and more accurate result. Yukai Yao et al. suggest that kmeans can work on small data and reduce the training and prediction time of SVM, because of no need of whole data set [16]. Jiaqi Wang, Xindong Wu and Chengqi Zhang [17] use SVM for business intelligence. But, SVM took more time for processing real BI application. They used KMeans, which provided higher response time with same accuracy. After that we performed SVM training for assigning data to different classes (clusters). After training phase, program generates .xml file of parameters and labels for data. This xml file is used in testing phase. In second phase we perform prediction on whole data set. Finally, we generate output image, which is classified image. For making it parallelize, we apply OpenMP on KMeans Clustering. Also we try OpenMP on svm training phase and svm prediction phase. Then we tested these different versions of programs on Intel i3, Intel i5 and Intel i7.

5 Experiment Results

We used different system for analysis of our program. We set up OpenCV in Visual Studio 2012 on different systems. We tested performance with different size of data, different number of threads, and different number of clusters with different accuracy and different iteration. We used KMeans clustering for creating labels, which assigned to training data sample. It gives better result than randomly generated labels. We can see the difference in Fig. 3 and Fig. 4. Here, Fig. 2 is source image, which is generated from text formats data.

International Journal of Scientific & Engineering Research Volume 7, Issue 5, May-2016
ISSN 2229-5518

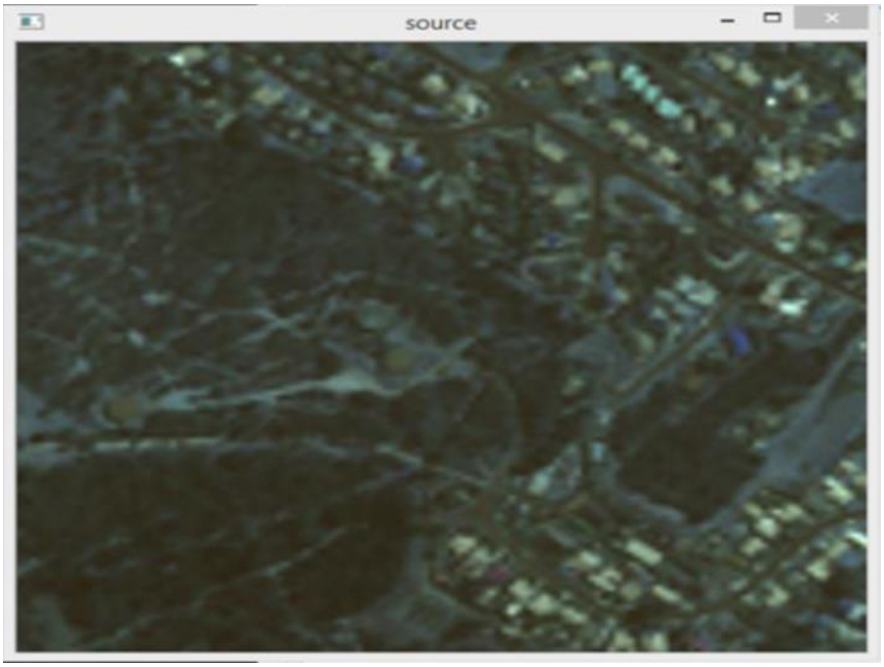

Fig. 2. Source Image (RGB)

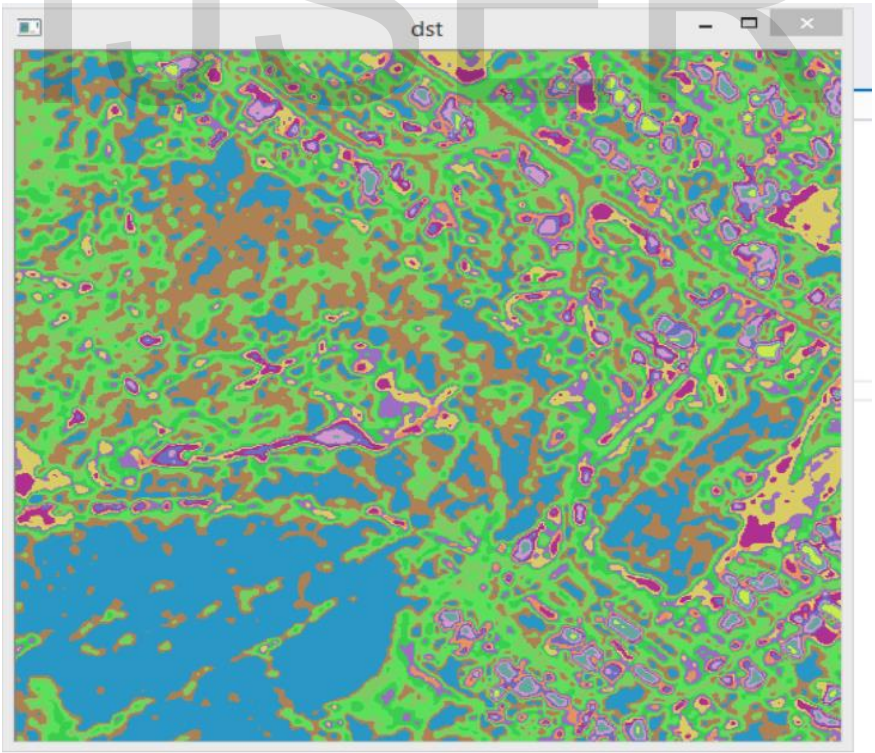

Fig. 3. SVM Classified Image with K-means

International Journal of Scientific & Engineering Research Volume 7, Issue 5, May-2016
ISSN 2229-5518

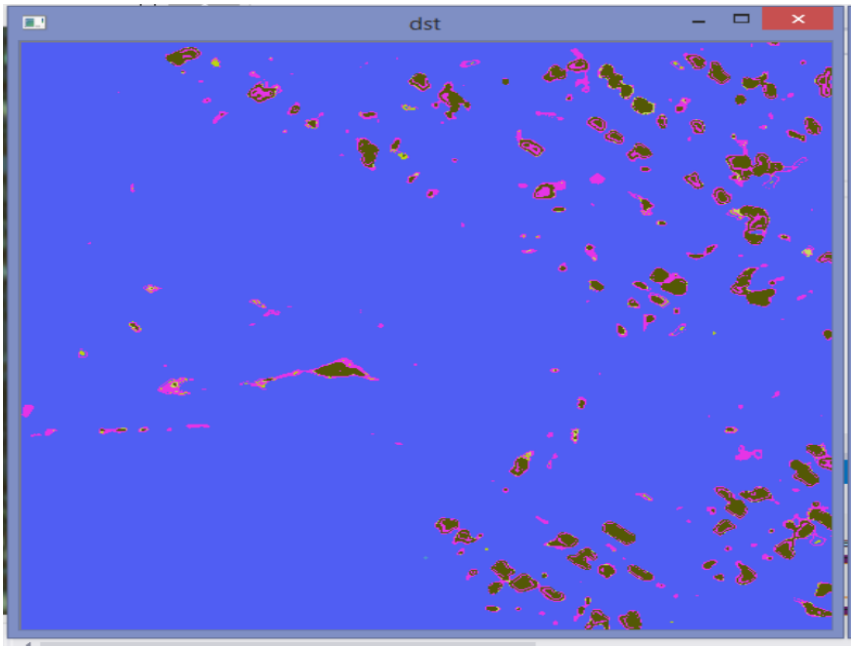

Fig. 4. SVM Classified Image without K-means

5.1 Performance on different CPU

We performed the tests on Intel i3, Intel i5 and Intel i7. We used 512 by 512 and 3000 by 3000 data (image). Here iter, eps and 512*512 & 3000*3000 are iteration of K-means, accuracy and image size, respectively. Small values of eps represent higher accuracy. Number of clusters (NC) is used in assigning the labels to training data in SVM training phase. And avg & max, which are average of different runs on different threads and maximum value among different runs respectively.

TABLE 1
EXECUTION TIME OF SVM ON INTEL I7 WITH DIFFERENT NUMBER OF THREADS (512*512)

SVM iter=1000 eps=.01 512*512									
NC	Thread	1	2	4	6	8	10	12	14
5	avg	6.143	3.98	3.936	4	4.0767	3.716	4.12	3.955
	max	6.4	4.23	4.1	4.15	4.51	3.86	4.24	4.16
7	avg	7.426	5.747	5.605	5.653	5.453	5.4	5.63	5.553
	max	7.67	5.98	6.06	5.85	5.91	5.71	6.04	5.64
10	avg	11.157	9.527	8.763	8.636	7.943	8.466	8.773	8.273
	max	12.03	10.5	9.12	9.01	8.36	8.84	9.16	8.76
13	avg	13.923	11.043	9.566	9.53	9.97	9.463	9.1367	9.51
	max	15.03	11.56	9.79	10.17	10.14	9.73	9.89	10.09
15	avg	16.75	11.747	10.63	10.827	10.05	10.663	10.503	10.627
	max	18.41	12.71	10.87	11.83	10.25	11.08	11.17	11.07
20	avg	21.123	13.997	11.587	12.503	12.097	11.59	11.72	11.487
	max	21.87	14.42	11.72	13.02	13.19	12.8	11.96	11.77

International Journal of Scientific & Engineering Research Volume 7, Issue 5, May-2016
ISSN 2229-5518

Table 1 represent the result for different cluster for different thread for 512 by 512 size data. From this table, we can depict that there is improvement in speedup, which is more than Intel i3 and Intel i7. Also because of more cache and more RAM memory than other two processors, Intel i7 give better performance. It is quad core processor. So it provides 8 threads and we can get more speedup than i3 & i5. In i3 and i5 processor, increment in speedup is limited up to 4 to 6 threads, while in Intel i7; it is up to 8 to 10 threads. Sometimes, there is decrement in speedup because of looping and other processes. Also synchronization limits the performance of multicore & multithreading technologies. Because of small size, there is not much difference with different threads.

Table 2 represents the result for different cluster for different thread for 3000 by 3000 size data. This table shows that i7 gives better performance for large data because of more threads and large memory and turbo boost also increase the frequency if processors when workloads increasing constantly on processors. Intel i7 gives 2 to 3 times better performance than Intel i3. Also here, RAM size is bigger for i7 than i3 and i5.

TABLE 2
EXECUTION TIME OF SVM ON INTEL I7 WITH DIFFERENT NUMBER OF THREADS (3000*3000)

SVM iter=1000 eps=.01 3000*3000									
NC	Thread	1	2	4	6	8	10	12	14
5	avg	113.98	80.49	75.666	79.313	72.193	73.82	75.23	74.63
	max	117.62	90.07	81.25	82.67	76.72	84.89	85.2	79.89
10	avg	251.65	159.81	153.04	159.24	137.26	127.64	145.17	132.40
	max	296.57	168.20	159.34	166.28	158.44	134.00	154.39	137.78
15	avg	340.65	220.10	184.47	199.91	193.34	187.71	180.51	180.56
	max	374.62	238.59	198.55	212.79	215.60	204.38	189.14	192.22
20	avg	482.08	316.24	252.54	285.47	254.19	248.41	226.66	234.92
	max	503.69	331.17	264.21	307.84	263.45	258.18	236.28	257.48

Figure 5 and 6 shows the chart for time (require for executing programs) versus threads for 512 by 512 size data and 3000 by 3000 size data, respectively. We can depict same results from both. Time for execution is decreasing rapidly from one thread to four threads. Then it slightly increased at 6 (six threads). Then it becomes linear in Figure 5. In Figure 6, execution time is same after 12 threads and it will increase for big number of threads.

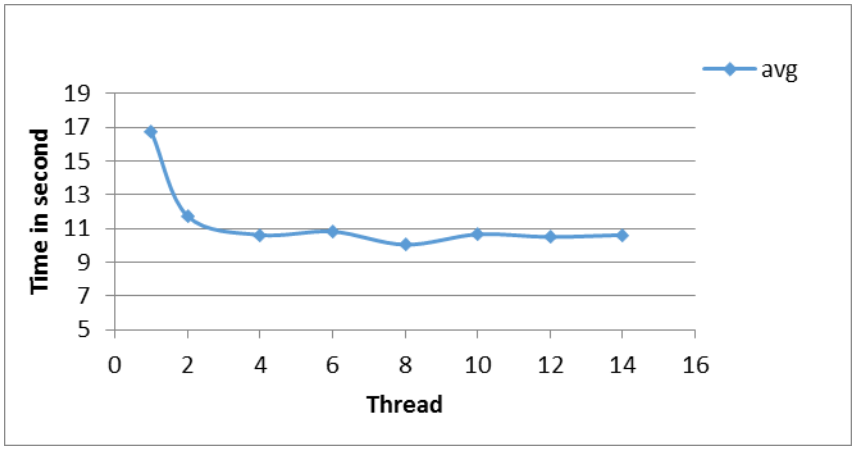

Fig. 5. Execution time for different number of threads (512*512)

International Journal of Scientific & Engineering Research Volume 7, Issue 5, May-2016
ISSN 2229-5518

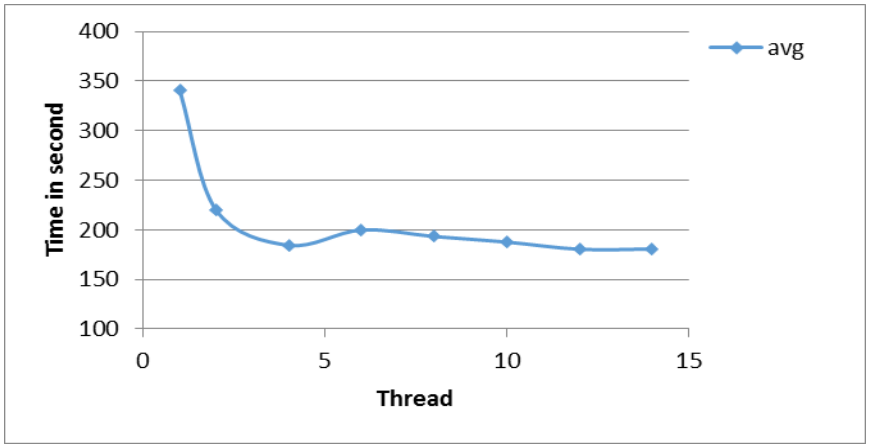

Fig. 6. Execution time for different number of threads (3000*3000)

Fig. 7. Speedup for different number of Clusters (SVM)

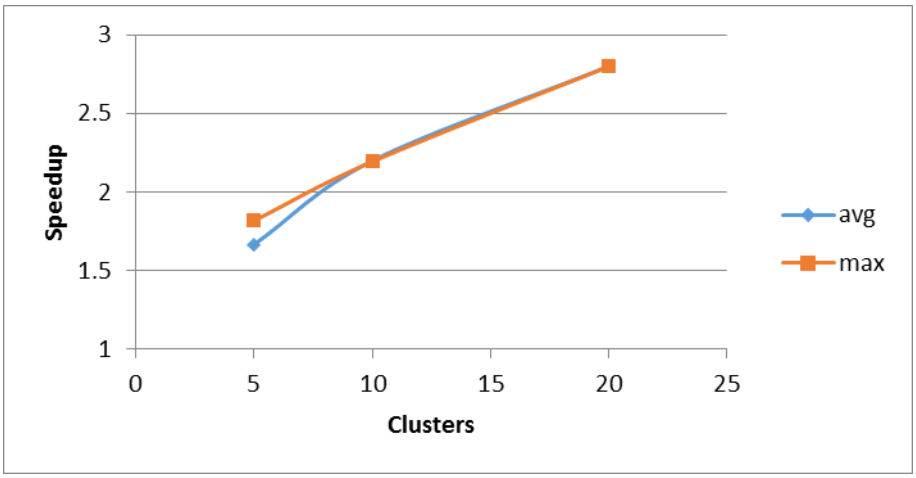

Fig. 8. Speedup for different number of Clusters (K-means)

International Journal of Scientific & Engineering Research Volume 7, Issue 5, May-2016
ISSN 2229-5518

We used the ratio from one thread to other threads like 1 to 2, 1 to 4, and so on. It is useful to analyze the speedup with different number of clusters as shown in Figure 7 and Figure 8.
From the Figure 7, we can say that speedup for SVM is rapidly decreasing from 1.7 to 1.3 for 5 clusters to 7 clusters. After that, speedup is gradually increasing up to 1.8 for 15 clusters. Than from 15 to 20 clusters, there is no more increment. From the Figure 8, we can say that speedup for K-means is gradually increasing up to 2.5 for 20 clusters from 1.6 for 5 clusters. It is better than SVM speedup for different number of clusters.

5.2 Performance with OpenMP

Here, we used OpenMP library for parallelization of K-means and SVM. And then I tested with 512 by 512 and 3000 by 3000 size images on Intel i3, i5 and i7. First we will see the results in table format then in graphical form.
Table 3, Table 4 and Table 5 shows the results for 512 by 512 size data. Because of small size, there is no more difference between two different versions of programs like SVM witout OpenMP and SVM with OpenMP version. Finally, we can say that, programs with OpenMP libraries takes more time for execution than programs without OpenMP. On Intel i3 and i5, there is approximately 60% more time require for programs with OpenMP.

TABLE 3
EXECUTION TIME FOR SVM AND K-MEANS WITH OPENMP & WITHOUT OpenMP on i3 (512 *512)

Thread	SVM	SVM with OpenMP	K-means	K-means with OpenMP
1	5.095	5.025	2.57	2.555
2	4.91	5.175	1.79	2.53
4	4.855	5.275	1.5	2.555
6	4.81	5.21	1.5105	2.555
8	4.89	5.35	1.49	2.55

TABLE 4
EXECUTION TIME FOR SVM AND K-MEANS WITH OPENMP & WITHOUT OpenMP on i5 (512 *512)

Thread	SVM	SVM with OpenMP	K-means	K-means with OpenMP
1	4.95	4.96	2.37	2.51
2	4.81	5.09	1.79	2.50
4	4.70	5.13	1.4	2.53
6	4.69	4.91	1.51	2.55
8	4.85	5.05	1.50	2.35

TABLE 5
EXECUTION TIME FOR SVM AND K-MEANS WITH OPENMP & WITHOUT OpenMP on i5 (512 *512)

Thread	SVM	SVM with OpenMP	K-means	K-means with OpenMP
1	1.99	2.145	1.17	1.17
2	1.97	1.945	0.89	1.17
4	2.12	2.11	0.84	1.195
6	1.955	2.065	0.87	1.21
8	1.99	2.11	0.775	1.21
10	1.99	2.11	0.785	1.215

Now, we see the result on Intel i7 for OpenMP version of SVM.
Table 6, Table 7 and Table 8 shows the results for 3000 by 3000 size data. There is no more difference between two different versions of programs for SVM and SVM with OpenMP. We can conclude that, SVM program with OpenMP libraries takes more 10 seconds time for execution than programs without OpenMP on Intel i3 and i5. And for i7 OpenMP version of SVM takes 4 to 6 seconds more than simple SVM. There is approximately 30 se-

International Journal of Scientific & Engineering Research Volume 7, Issue 5, May-2016
ISSN 2229-5518

conds more time requiring for K-means program with OpenMP means.
on i3 and i5, while i7 takes more 10 to 13 seconds than simple K-

TABLE 6
EXECUTION TIME FOR SVM AND K-MEANS WITH OPENMP & WITHOUT
OpenMP on i3 (3000*3000)

Thread	SVM	SVM with OpenMP	K-means	K-means with OpenMP
1	57.15	58.78	87.7635	88.545
2	51.855	60	61.895	88.03
4	50.63	58.94	52.37	90.91
6	49.95	59.275	52.03	87.915
8	49.935	59.415	51.445	87

TABLE 7
EXECUTION TIME FOR SVM AND K-MEANS WITH OPENMP & WITHOUT
OpenMP on i5 (3000*3000)

Thread	SVM	SVM with OpenMP	K-means	K-means with OpenMP
1	56.63	58.26	85.243	87.025
2	51.335	59.48	60.375	86.51
4	50.11	58.42	50.85	89.39
6	49.43	58.755	50.51	86.395
8	49.415	58.895	49.925	85.48

TABLE 8
EXECUTION TIME FOR SVM AND K-MEANS WITH OPENMP & WITHOUT
OpenMP on i7 (3000*3000)

Thread	SVM	SVM with OpenMP	K-means	K-means with OpenMP
1	23.53333	23.61	40.28	40.74
2	20.53	23.26	30.01	40.76
4	19.61867	23.085	27.912	40.355
6	18.27667	25.91333	30	40.595
8	18.33	23.33	27.095	40.444
10	18.895	26	27.15	40.50

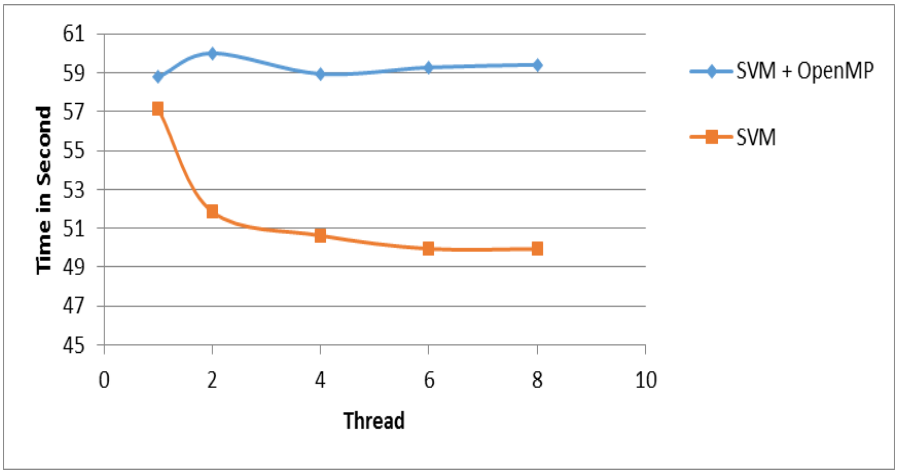

Fig. 9. Execution time for SVM without OpenMP & SVM with OpenMP

International Journal of Scientific & Engineering Research Volume 7, Issue 5, May-2016
ISSN 2229-5518

The chart in Figure 9, reveal that OpenMP decrease the speedup of SVM in OpenCV. Without OpenMP, SVM takes 50 seconds on 8 threads, while with OpenMP it takes 59 seconds. Also SVM for less number of classes cannot get more benefit of multicore processors in OpenCV. Use of OpenMP in OpenCV programs explicitly increases the execution time. There is parallel loop in OpenCV source files. Because of that explicit use of OpenMP is not beneficial for OpenCV programs.

6. Conclusion

Nowadays we have bigger size images and video because of advanced technologies. For making faster processing on those data different parallelization concepts are used. The SVM is mainly machine learning method and widely used in image processing because it provides better accuracy as time passed. Also in addition, algorithm with OpenMP and CUDA give faster result than conventional algorithm. Parallel algorithms with CUDA give higher result than OpenMP. But parallel programming with OpenMP is less complex than CUDA programming and OpenMP is available for most processors, which are used nowadays.

This paper presented some important efforts to apply SVM and k-means for image processing. For optimize and reduce run time of SVM, we used OpenMP. We tested on different processors. From obtained results, we concluded that KMeans and SVM give better performance for large number of clusters than small number of clusters on multicore processors. Because of implicit parallelization, OpenMP increase the execution time up to 20% to 40% for SVM programs in OpenCV.

For taking the benefit of OpenMP in OpenCV, we need to make more changes in source file. Also we can use other parallelization libraries like Intel TBB, CUDA or hybrid libraries like MPI and OpenMP or CUDA and OpenMP.

References

[1] Support Vector Machines: SVM www.support-vector-machines.org/

[2] Pooja Kamavisdar, Sonam Saluja, Sonu Agrawal, "A Survey on Image Classification Approaches and Techniques", IJARCCE, Vol. 2, Jan 2013

[3] http://docs.opencv.org/master/d1/d73/tutorial_introduction_to_svm.html#gsc.tab=0

[4] Yukai Yao, Yang Liu, Yongqing Yu, Hong Xu, Weiming Lv, Zhao Li, Xiaoyun Chen, "K-SVM: An Effective SVM Algorithm Based on K-means Clustering", Journal of Computers, 2013

[5] Greg Slabaugh, Richard Boyes, Xiaoyun Yang, "Multicore Image Processing with OpenMP", IEEE, March 2010

[6] OpenMP Architecture Review Board. OpenMP specifications. http://openmp.org/wp/openmp-specifications/

[7] Kun Tan, Junpeng Zhang, Qian Du, Xuesong Wang, "GPU Parallel Implementation of Support Vector Machines for Hyperspectral Image Classification", IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2015

[8] Priya P. Sajan, S.S. Kumar, "Experimental Study on Parallel Wavelet Based Image Segmentation Using Openmp", International Conference on Control, Instrumentation, Communication and Computational Technologies, IEEE, 2014

[9] Rashmi C, Dr.Vinay K, Dr. Hemantha Kumar G, "An Efficient Multithreading Approach for Localization of Face Parts", IEEE International Advance Computing Conference, 2015

[10] Pengfei Chang, Zhuo Bi, and Yiyong Feng, "Parallel SMO Algorithm Implementation Based on OpenMP", IEEE International Conference on System Science and Engineering (ICSSE), 2014

[11] Suli Zhang, Haoran Hu, Xin Pan, "Parallelized Dilate Algorithm for Remote Sensing Image", The Scientific Journal, May 2014

[12] Harshad B. Prajapati, Dr. Sanjay K. Vij, "Analytical study of parallel and distributed image processing", International Conference on Image Information Processing (ICIIP), 2011

[13] Lan Xiaowen, "Research on Multi-Core PC Parallel Computation Based on OpenMP", IJMUE, 2014

[14] P. E. Hadjidoukas, V. V. Dimakopoulos, "A high performance face detection system using OpenMP", Concurrency and Computation: Practice and Experience, Vol. 21, No. 15, Oct. 2009

[15] Liuyang Fang, Mi Wang, Deren Li, and Jun Pan, "MOC-Based Parallel Preprocessing of ZY-3 Satellite Images", IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, VOL.12, NO. 2, FEBRUARY 2015

[16] Yukai Yao, Yang Liu, Yongqing Yu, Hong Xu, Weiming Lv, Zhao Li, Xiaoyun Chen, "K-SVM: An Effective SVM Algorithm Based on K-means Clustering", Journal of Computers, 2013

[17] Jiaqi Wang, Xindong Wu, Chengqi Zhang, "Support vector machines based on K-means clustering for real-time business intelligence systems", Int. J. Business Intelligence and Data Mining, 2005.

[18] Sumit Patel, Dr. M. B. Potdar, Bhadreshsinh Gohil, "A Survey on Image Processing Techniques with OpenMP", IJEDR, Dec 2015.