Resource Optimisation using Multithreading in Support Vector Machine

Wong Soon Fook[1]
Faculty of Information Science and Technology
Universiti Kebangsaan Malaysia

Abdul Hadi Abd Rahman[2]*, Nor Samsiah Sani[3]
Afzan Adam[4]
Center for Artificial Intelligence Technology (CAIT)
Universiti Kebangsaan Malaysia

***Abstract*—Image processing is one of the most important features for vision-based robotic and being used in various applications to increase productivity. Various researchers reported issues computation problem to detect objects in low cost device such as vision-based robotic car. In the fast-paced development of technology, a system that runs automatically with the right results is essential to the completion of a job. This study aims to propose an effective multithreading for road sign recognition. We implemented multithreading algorithm for train and detector processes in SVM to utilise the multicore CPU and evaluate in various condition on by a Raspberry Pi platform. It aims to solve the real-time computation issue using Pi camera. Experimental results show significant improvement of performance to the detection accuracy. In conclusion multithreading significantly improve the detection performance using Raspberry Pi processors with various image resolution and number of SVM model.**

***Keywords*—Robot vision; recognition; multithreading; real-time**

I. Introduction

Image processing is one of the most important features for vision-based robotic and being used in various applications to increase productivity. One of the interesting topic is object recognition which has been evolved drastically. In robotic contexts, the ability to understand the object helps robot to make accurate and better decision [1][2]. However due to large resource consumption for computation, multithreading method are one the way to optimize using multi-tasking process and fasten the computation in real-time application. Timing is an important factor in image processing because the delay in time or delivery of an image template would cause many issues in the final decision. This lead to adoption the concept of multithreading in low cost computing device such as Raspberry Pi so that the results of recognition are accurate.

In addition, development of intelligent car robot is also a symbol of modernization and development that is rapidly changing [3]. Each features of cars and transportation are created to help in the comfort and safety of everyone. As such, this study focus on about vision-based robotic cars with improvements in multithreading and image processing. A multithreading algorithm is implemented to detect images such as signage with the addition of multithreading to the system for better performance. This method is applied on a machine learning algorithm called Support Vector Machine as for image training and detection process [4][5]. Evaluations of its performance focused on variation of input, model and resource optimization.

This paper is organized in five sections. Section I provides an overview of issues and research gap. Section II presents the related work on object recognition and multithreading. Section III describes the research methodology implemented in this study. Section IV presents the experimental result and discussion on the finding. Finally, Section V concludes the impact of this study.

II. Related Works

The use of Intelligent Robotic Car is very efficient when the robot itself will move autonomously as the robot understands each sign. Furthermore, it responds to the detected sign without requiring the user to move it. However, various researchers reported issues computation problem to detect objects in low cost device such as Smart Car Robot [6][7]. This is due to the Raspberry Pi has four cores but only the use of a single core can be achieved. The use of this single core resulted in the performance of the Raspberry Pi slowing down for the Pi camera detecting the sign [8]. Images that can be detected using the Support Vector Machine algorithm are also limited to fast detection when only a single core is used resulting in performance on the system. The detection using the Pi camera is slower when more images are stored as SVM models [9] [3].

The simplest type of multithreading occurs when a thread runs until it is blocked by an event that usually creates a long latency [10]. Such a stop may be due to the cache having to access the external chip memory, which may take hundreds of CPU cycles for the data to be returned. Instead of waiting for a stop to be completed, the threading processor will switch the implementation to another thread that is ready to run. Only when the data for the previous thread has arrived, will it allow the previous data to be placed on the standby thread list. The purpose of multithreading is to remove all interrupted data dependencies from the implementation pipeline [11,12]. Because one thread is independent of another, there is a possibility of a single instruction in a pipeline that requires output from a longer direction in the planning. Conceptually, it is similar to the primitive multitasking used in operating systems; The analogy is that the time given to each active thread is a CPU cycle. The most advanced type of multithreading applies to superscalar processors. Whereas normal superscalar processors issue multiple commands from one thread per CPU cycle, in simultaneous multithreading

*Corresponding Author

(SMT) the superscalar processor can issue commands from multiple threads per CPU cycle. Realizing that any single thread has a limited amount of directive parallelism, this type of multithreading attempts to exploit the parallelism found in the various threads to minimize the rest associated with unused issue slots.

The objectives of this study focused on further evaluation of signal processing using the Pi camera with several variables to test to improve system performance. Furthermore, comparison of single core based Raspberry Pi with multicore via multithreading so that CPU usage and Raspberry Pi memory are analysed.

III. Research Methodology

This study is divided into five phases which contains collection of data, annotation, training, detection using SVM and improvement using multithreading procedure. In this phase we considered issues when the increase in the number of images in each SVM model for detection by a Pi camera significantly improves performance to the detection accuracy decreases. During the process of running on the device, the use of 1 core on the Raspberry Pi greatly reduced the memory usage which led to the loss of the stored image because lack of support and storage of multiple images which delayed its performance.

A. Data Collection

This project is about the detection of signage so the collection of signage images is from the source https://github.com/Moataz-E/deeplearning-traffic-signs. Each description used has a different information. The images collected are from a range of resolutions to be set to four resolutions of 160x128, 240x192, 640,480 and 1296x736. Increasing the resolution at each detection will test the system's ability to function efficiently. Performance data during benchmark detection testing were collected and reported for performance evaluation using selected attribute such as resolution and image amount.

B. Support Vector Machine

In machine learning, support vector machine (SVM) is a learning models which integrates learning algorithms related to data analysis used for classification and regression analysis. Since a set of training examples, each labeled as belonging to one or the other of two categories, SVM training algorithms build models that provide new examples to one category or another, they become binary linear classifiers that are non-existent (though methods like scaling exist to use SVM in probabilistic classification settings). The SVM model is a representation of the samples as points in space, mapped so that the separate categories are divided into as wide a gap as possible as shown in Fig. 1. The new examples are then mapped into the same space and predicted to become categories based on the sides of the gap.

All training image were annotated to set the size limit to the image to be detected. It aims to classify images by dividing hyperplanes into non-linear datasets. Classification of each object by maximizing the margin distance so that the data points can be classified more confidently. SVM is one of the low computation machine learning algorithms which is suitable due to the limitations of the Raspberry Pi in handling high demand process and algorithmic demand.

C. Multithreading

This study focuses more on internal performance than on external performance, which is more on Raspberry Pi's performance in the ability to carry out signage detection with large picture storage and higher resolution images. Pre evaluation were done for each SVM processes to trace the high computation process for multithreading [13,14]. The sign-on process is used to monitor and logged the performance of the Raspberry Pi system for pre and post multithreading evaluation.

In this phase, the detection of the trained signage using the SVM algorithm. Signal detection using the Pi camera and when the trained sign image is detected, green, red, blue or white frames will appear around the image known as the image marker for detected image. In computer architecture, multithreading is the ability of a central processing unit (CPU) (or single core in a multi-core processor) to execute multiple processes or threads simultaneously supported by operating systems. This approach is different from multiprocessing. In multithreaded applications, processes and threads share single or multiple core sources, including computing units, CPU caches, and lookaside translation buffers (TLB). A multiprocessing system includes multiple complete processing units in one or more cores, multithreading is intended to enhance single core use by using thread-level parallelism, as well as command-level parallelism. Because the two techniques complement each other, they are sometimes combined in a multithreading CPU system and with a multi-core CPU.

The multithreading algorithm as in Fig. 2 is deployed on existing coding during model detection process. In pre-evaluation, the image streamed from the Pi camera show lagging issues but not at the capture stage, annotate the image and train the image to the SVM model. This is caused by our very large SVM model files with a very large number of images will cause our computer performance and high CPU memory usage. From a coding standpoint, the use of just one thread per process in the fourth coding which results in overloading of only one CPU memory will result in the accuracy of the tracking results being dropped while we can access all four cores on the Raspberry Pi 3B+ to split memory usage CPU evenly. Due to memory limitations on only one CPU, implementation of multithreading alternatives should improve the tracking performance in real-time.

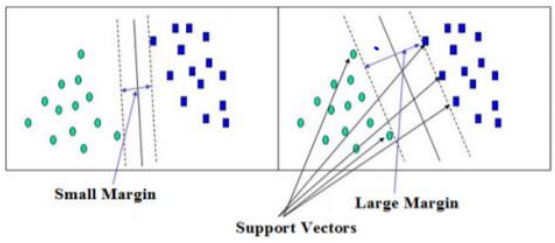

Fig. 1. Separation between Small and Large Margins of SVM.

```
initiate PiCamera(set fps)
image to RGB
set myPath=[]
for each file in myPath
 append learningoutput.svm
end for
set thread = []
for each oneDetector in detectors
 thread = svmDetector
 thread.start()
 thread.append()
end for
```

Fig. 2. Multithreading in Support Vector Machine Algorithm Pseudocode.

IV. Result and Discussion

A. Comparison of CPU Memory usage in Frame Per Second

Fig. 3 shows the original code executed with the performance monitoring taken during the execution of the code. Currently only two processes are running - SVM model detection and performance monitoring that can be seen in the above diagram. From the system monitoring can look at process identifier number 842, CPU usage was 85.3% with 9.4% memory by the coding. On the record it can be stated that 4 CPUs are used. From there the core usage guarantees by looking at the number on us is the usage in the Raspberry Pi core. It can be seen that only one core is used here and subsequent tracking is still ongoing.

In Fig. 4, the output of coding added with multithreading programming is presented. Originally, only one running process is SVM model tracking code. With the use of multithreading, it can be seen that the optimization of resource is achieved show by the usage of the 4 CPU cores in an evenly distributed processes. This is due to every 1 SVM model uses 1 thread to run the process from the original code compared to usage of single thread to run the entire SVM model. Usage of less than 50 with decrement of memory usage from 9.4% to 7.5%. The use of multithreading shows an improvement in computing performance.

Table I shows a graph of thread usage on FPS performance. The first experimental test used single thread computation with a recorded 10 frame per second followed by the use of 2 threads resulted in an increased FPS of 15. The FPS in an optimized solution using all 4 threads improved to 30 due to the system's reluctance to run every single thread, containing the process as a separate thread.

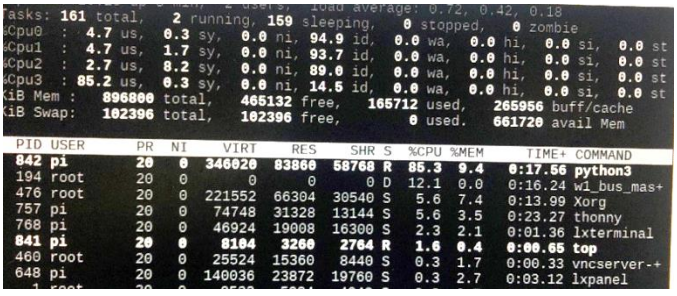

Fig. 3. Performance on Raspberry Pi without Multithreading.

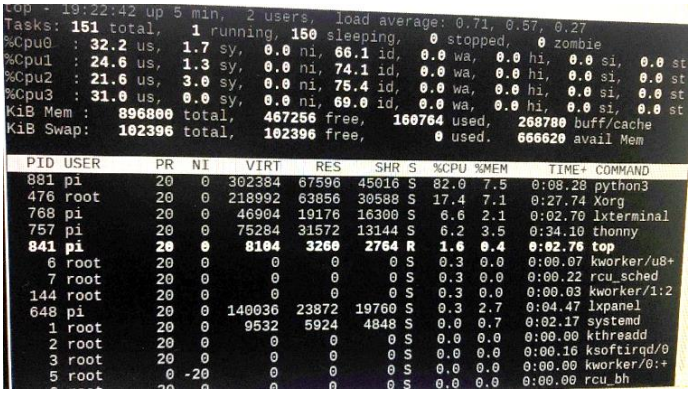

Fig. 4. Performance on Raspberry Pi with Multithreading on 4 Processor.

TABLE I. Performance on Raspberry Pi with Multithreading

Parameters Thread Used	**Without Multithreading** ***1***	**Multithreading**	
		2	***4***
FPS	10	15	30
Memory	9.4%	7.5%	7.6%
CPU Usage	85.3%	86.2%	80.3%

B. Comparison of Resolution with use of Multithreading on Memory and CPU

Table II shows some of the resolutions used to run tests to evaluate performance I various resolution conditions. The evaluations considered important parameters such as time, memory and CPU Usage which are presented in Table III. The results indicate an improvement over time and memory in various resolutions.

Fig. 5 shows the graph increasing with time as resolution increases. The difference between using a thread and not using a thread is a small amount of time recorded but improvements have been made to the system. Time was recorded according to the 5 recorded pictures and the last time the fifth picture was taken to draw the graph. It can be seen that there is a slight increase in graphs using threading compared to no threading.

TABLE II. Resolution Specifications

Resolution	**Aspect Ratio**	**Frame Rate**	**FoV**
160x128	4:3	30fps	*PARTIAL*
240x192	4:3	49fps	*PARTIAL*
640x480	4:3	42.1-60fps	*FULL*
1296x736	16:9	1-49fps	*FULL*

TABLE III. Performance using Thread with Various Resolutions

Parameters	**Resolution**		
	160x128	240x192	640x480
Time (ms)	2.39	4.66	27.33
Memory	7.4%	7.7%	12.6%
CPU Usage	74.5%	93.2%	99.7%

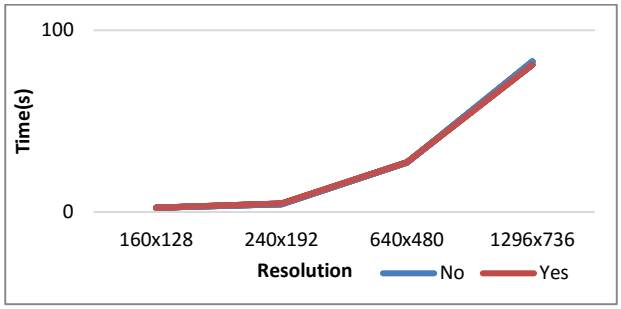

Fig. 5. Comparison of Thread usage with Time and Resolution.

C. Comparison of Various SVM Models and Images using Multithreading

Table IV shows the use of threading for image processing by increasing the image from 10 to 50 and improving the trained SVM model. Increased processing time for uniform images in 1.3 and 5 models. For the 30 images, time increased drastically on the third model and gradually increased upon the fifth model. For the 50 images, the time increases parallel to capital 1 to 5. Visible in the 5th capital with 50 images using threading is faster than the non-threading detection with 3 models and 30 images per model which is 23.7 tracking time.

TABLE IV. COMPARISON OF PERFORMANCE USING YARN WITH VARIABLE NUMBER OF SVM MODELS

Model	**Total Images**		
	10	**30**	***50***
1	2.5 s	4.3 s	5.2 s
3	4.6 s	11.8 s	16.5 s
5	7.4 s	14.7 s	23.7 s

V. CONCLUSION

The development of the intelligent robotic system aims to improve the computing performance by optimizing the resources in a Raspberry Pi. Experimental results show a significant improvement achieved using multithreading in SVM processes. Based on the research conducted, there are several suggestions for further improvements, such as deep learning and algorithms like Fractal or any other machine learning approach such as RNN. In conclusion, the study intended to benefit road users so that they can receive information about road signage with high performance.

ACKNOWLEDGMENT

The authors want to thank the University Kebangsaan Malaysia for supporting and funding this research, grant code: GGPM-2017-040.

REFERENCES

[1] F. F. Saad Mohmad Saad Ismail, S.N.S Abdullah, “Detection and recognition via adaptive binarization and fuzzy clustering,” Pertanika J. Sci. Technol., vol. 27, no. 4, pp. 1759–1781, 2019.

[2] G. Alipoor and E. Samadi, “Robust Gender Identification using EMD-Based Cepstral Features,” Asia-Pacific Journal of. Information Technology and Multimedia., vol. 07, no. 01, pp. 71–81, Jun. 2018.

[3] N.F.A Zainal, R. Din, M.F. Nasrudin, S. Abdullah, A.H.A Rahman, S.N.S Abdullah, K.A.Z. Ariffin, S.M. Jaafar, N.A.A Majid. (2018). Robotic Prototype And Module Specification For Increasing The Interest Of Malaysian Students In Stem Education. - International Journal Of Engineering And Technology (Uae).

[4] K. Vinothini and S. Jayanthy, “Road Sign Recognition System for Autonomous Vehicle using Raspberry Pi,” in 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), 2019, pp. 78–83.

[5] C. Day, L. McEachen, A. Khan, S. Sharma, and G. Masala, “Pedestrian Recognition and Obstacle Avoidance for Autonomous Vehicles Using Raspberry Pi,” 2020, pp. 51–69.

[6] V. Patchava, H. B. Kandala, and P. R. Babu, “A Smart Home Automation technique with Raspberry Pi using IoT,” in 2015 International Conference on Smart Sensors and Systems (IC-SSS), 2015, pp. 1–4.

[7] E. Bilgin and S. Robila, “Road sign recognition system on Raspberry Pi,” in 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), 2016, pp. 1–5.

[8] S. Brahmbhatt, “Embedded Computer Vision: Running OpenCV Programs on the Raspberry Pi,” in Practical OpenCV, Berkeley, CA: Apress, 2013, pp. 201–218.

[9] M. R. Rizqullah, A. R. Anom Besari, I. Kurnianto Wibowo, R. Setiawan, and D. Agata, “Design and Implementation of Middleware System for IoT Devices based on Raspberry Pi,” in 2018 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC), 2018, pp. 229–234.

[10] D. R. Rinku and M. Asha Rani, “Analysis of multi-threading time metric on single and multi-core CPUs with Matrix Multiplication,” in 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), 2017, pp. 152–155.

[11] W. F. Abaya, J. Basa, M. Sy, A. C. Abad, and E. P. Dadios, “Low cost smart security camera with night vision capability using Raspberry Pi and OpenCV,” in 2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 2014, pp. 1–6.

[12] Azmi, I., Shafei, M. S., Nasrudin, M. F., Sani, N. S., & Abd Rahman , A. H. . ArUcoRSV: Robot localisation using artificial marker. In J-H. Kim, H. Myung, & S-M. Lee (Eds.), Robot Intelligence Technology and Applications - 6th International Conference, RiTA 2018, Springer Verlag, 2019, pp. 189-198.

[13] M. M. William et al., "Traffic Signs Detection and Recognition System using Deep Learning," 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 2019, pp. 160-166.

[14] Zuraini Othman, Azizi Abdullah, Anton Satria Prabuwono. (2018). Iris Localization Algorithm Using Region Growing and Support Vector Machine. - Advanced Science Letters. 1005-1011.