Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.023953
Article

Moving Object Detection and Tracking Algorithm Using Hybrid Decomposition Parallel Processing

M. Gomathy Nayagam[1,*], K. Ramar[2], K. Venkatesh[3] and S. P. Raja[4]

[1]Department of Computer Science and Engineering, Ramco Institute of Technology, Rajapalayam, 626117, India
[2]Department of Computer Science and Engineering, R.M.K. College of Engineering and Technology, Chennai, 601206, India
[3]Department of Computer Science and Engineering, Veltech Rangarajan Dr Sagunthala R&D Institute of Science and Technology, Chennai, 600062, India
[4]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, 632014, Tamilnadu, India
*Corresponding Author: M. Gomathy Nayagam. Email: m.g.nayagam@gmail.com
Received: 28 September 2021; Accepted: 17 December 2021

Abstract: Moving object detection, classification and tracking are more crucial and challenging task in most of the computer vision and machine vision applications such as robot navigation, human behavior analysis, traffic flow analysis and etc. However, most of object detection and tracking algorithms are not suitable for real time processing and causes slower processing speed due to the processing and analyzing of high resolution video from high-end multiple cameras. It requires more computation and storage. To address the aforementioned problem, this paper proposes a way of parallel processing of temporal frame differencing algorithm for object detection and contour tracking using the mixture of functional and domain decomposition parallel processing techniques. It has two main contributions. First, steps of frame differencing are parallelized using functional decomposition technique. Second, the processing of frames in each steps of frame differencing is again parallelized using domain decomposition technique. Finally, the performance is evaluated in Aneka Private Cloud platform and which yields to detect and track the object very swiftly and accurately.

Keywords: Temporal frame differencing based object detection; functional and domain decomposition parallel processing techniques contour tracking and cloud computing

1 Introduction

In automated video surveillance system [1], the task of detection and tracking of moving object is a important part and it can be done be different approaches. But it does not have common approaches for all applications. Background subtraction is one of the most popular and robust method for moving object detection [2,3]. But there is a challenge in background subtraction due to dynamic background, illumination changes and etc. [4]. The temporal difference method is used in [5] where they proposed reliable foreground segmentation algorithm which combines temporal image analysis with reference background image. It is suitable to adapt for background change in illumination. In [6], they extract

moving object based on temporal differencing mechanism and they used Normalized Cross Correlation (NCC) technique to remove Ghost and shadow effect during temporal frame differencing mechanism. Smart cameras that perform real time video analysis are recent trends in surveillance system [7].

Due to the amount of data that needs to be processed for providing high accuracy in real time video surveillance systems, most of the algorithms are having difficult and time consuming characteristics in nature. If we perform video analysis in serial processor, it requires more computation time and uses significant amount of energy which bounds the application of such algorithm in live video analysis.

GPU are popular platform for parallel processing of objection and tracking [8] using software such as openCL and CUDA. Due to the physical constraints and high energy consumption of GPU, there is a need to modify the algorithm for the particular embedded GPU chipset. Otherwise, it results in suboptimal performance in such GPU chipsets.

Another way is the implementation of parallel algorithm on embedded systems which is based on digital signal processor (DSP) and field programmable gate arrays (FPGA) [9]. Since, this system have low number of computing units, this solution is suitable only for low video resolution and low frame rate video processing.

Cloud computing is one of the platform for implementing parallelization of moving object detection and tracking algorithm without any modification for optimization and also does not require any specialized hardware. So, implementing parallelization of moving object detection and tracking algorithm in cloud platform is more efficient and cost effective [10–12].

In this paper, parallel version of temporal frame difference based moving object detection with contour tracking is presented. In order to achieve functional decomposition, there are three master threads are created in this proposed work. First one is responsible for motion detection using temporal frame differencing, second one is responsible for noise removal in foreground image and third one is responsible for blob detection and track the individual objects. The moving object detection thread is again divided into child thread for inherent parallelization of pixel by pixel subtraction operation. The work will outperforms to detect multiple objects at same time with high speed with good accuracy.

Remaining of the paper is organized as follows: Section 2 surveys the relevant work done using cloud computing and other technologies. Section 3 describes the proposed work. Section 4 describes results and performance evaluation. Section 5 discusses conclusion and future work.

2 Proposed Work

The proposed cloud computing based rapid object detection and tracking system is elucidated in this section.

2.1 Overview of the Work

The main aim of this article is to present cloud computing based rapid object detection and tracking system. Most of the object detection and tracking algorithm requires more computational and storage for processing video. The cloud computing will provide environment which has more computational and storage requirements to process the video for detecting and tracking object in it with very rapidly. The following Fig. 1 shows the architecture of cloud based moving object detection and tracking system in video. The proposed system assumes the camera is fixed in the particular surveillance area with fairly wide angle of view.

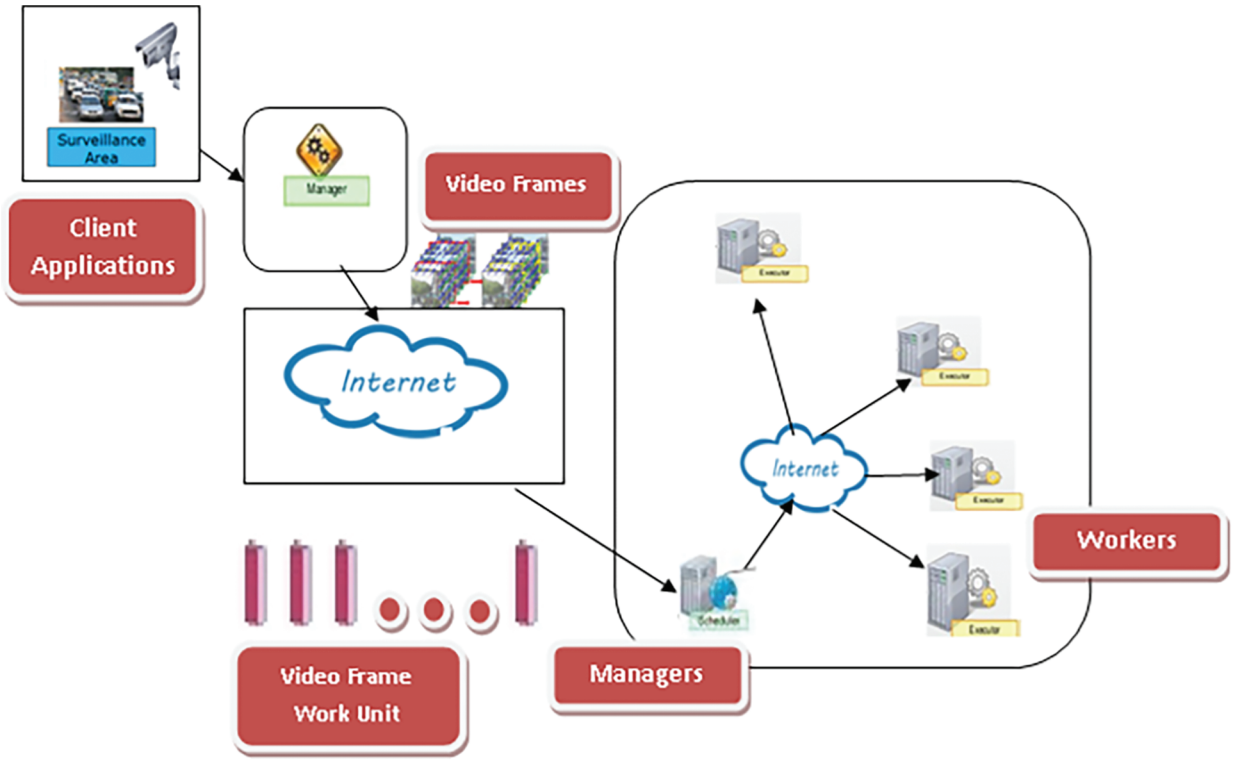


Figure 1: Architecture of cloud based moving object detection and tracking system

The frequently captured video file is transferred to cloud environment through GUI and cloud manager where video are sliced into frames and each frames are considered as a work unit of cloud environment. Each of the work unit is distributed to executor by using cloud scheduler. The system components of proposed system architecture includes: video capture, motion detection, Image Filtering, Blob detection, contour detection, updatation and tracking.

In order to parallelize these components, the proposed system architecture uses functional decomposition based parallelism. Hence, each functional component of temporal object detection and tracking algorithms are executed parallelly by the worker node. Again, these functional decomposed components are internally parallelized using inherent sequential parallelizing mechanism which is executed by the individual workers nodes. The Fig. 2 shows the functional decomposition based parallelizing mechanism to execute the functional components of object detection and tracking algorithm parallely.

2.2 Video Capture

The input can be taken in two ways using camera or giving path for stored video file which are captured from camera sensors. Live video streams need to be processed. Once the video captured which are converted in to frames and processed. Frames in videos are sequence of images which is a representation of two dimensional images as a finite set of digital values called pixel [13–15] in digital computer. Digital images are represented as 2D matrices which contains finite set of elements px, y which is the color information for the corresponding pixel in the image at the given location x, y.

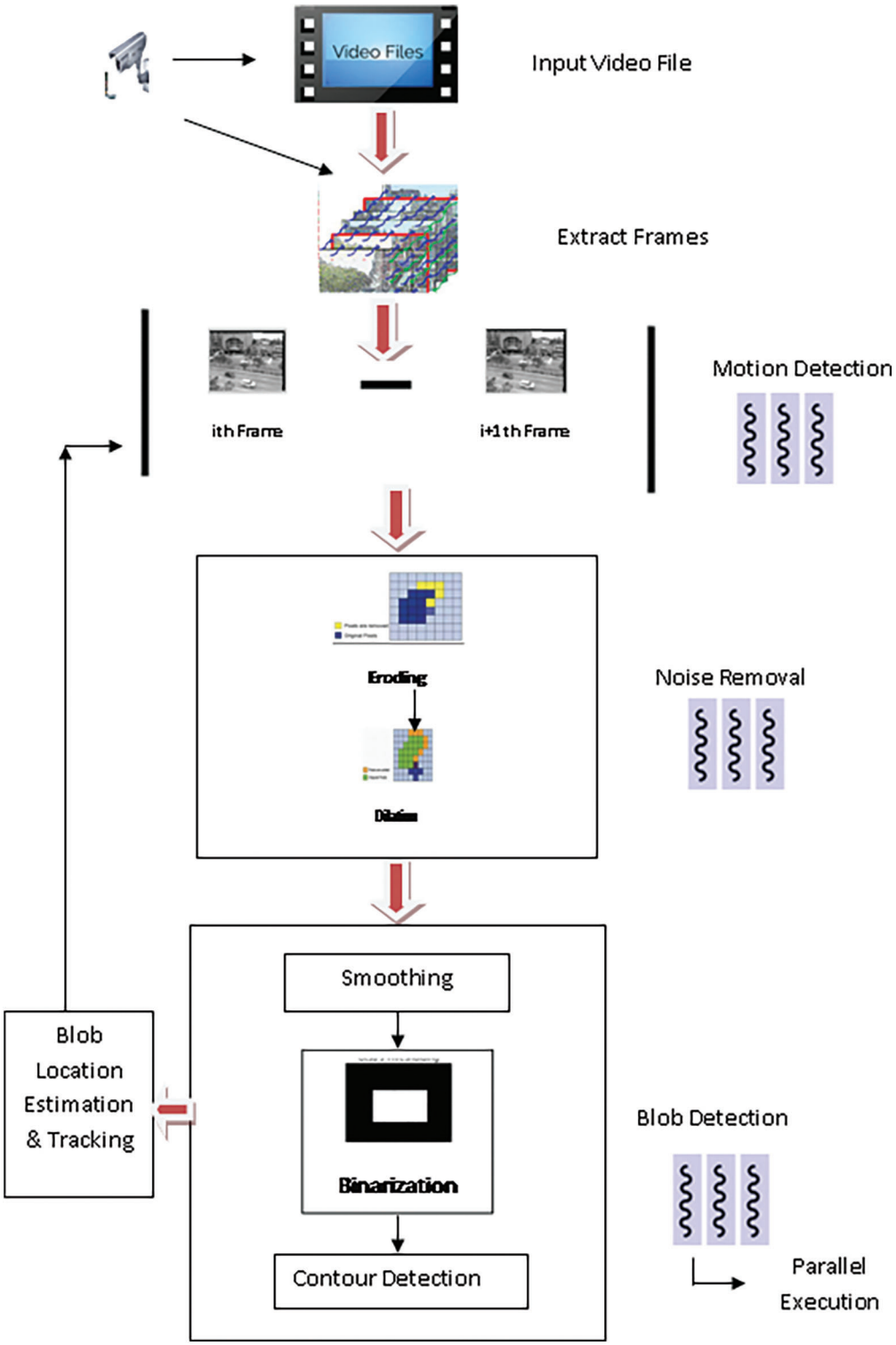


Figure 2: Functional decomposition based parallel execution for temporal frame differencing

2.3 Motion Detection Using Temporal Frame Differencing

In this work, detection of moving object using temporal frames differencing [16] which is the simplest background subtraction techniques. Also, it is a non-recursive method; hence there is no need to maintain the history of frames in buffer. So, memory requirement for this algorithm is low compared to other existing algorithms. In this method, current frame's pixel values are subtracted with previous frames pixel value. Based on the thresholding value, moving region of the image is extracted and identifies the both foreground and background [17]. The resultant of absolute pixel value is zero indicates, there is no moving object or non-zero value of this pixel value indicates there was a moving object in frames.

According to the method describe above, the difference between the frame at t and frame at t−1 is determined as follows [18,19]:

$$D_k = |I_{x,y,t} - I_{x,y,t-1}| \tag{1}$$

For adopting the changes in background illumination [19,20], the reference image is updated at each iteration as follows:

$$B_{t+1} = (1 - \alpha)B_t + \alpha I_{x,y,t} \tag{2}$$

The pixel value of frames/image is represented as a matrix and matrix subtraction operations carried out for motion detection. The size of image is divided into smaller kernel like N × N matrix and each kernel of adjacent frame pixels are created as a thread which are executed in different cloud executor using inherently sequential decomposition mechanism. The Fig. 3 shows the inherently sequential based functional domain decomposition for temporal frame differencing.

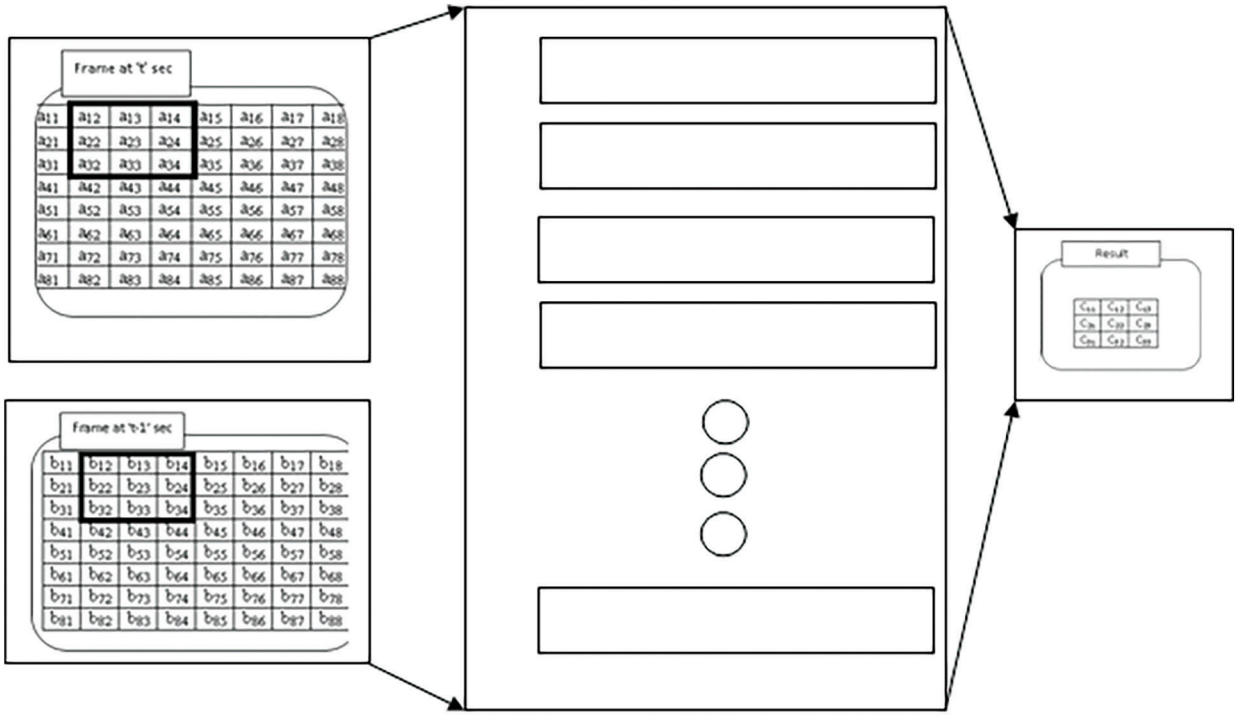


Figure 3: Inherently sequential based functional domain decomposition for temporal frame differencing

2.4 Noise Removal

Some of the background frames have spurious movement of object. It leads to mistaken in detect and track the object. So, this noise causing elements have to be filtered. For this purpose, this work used two basic morphological operations viz erosion and dilation. The basic effect of dilation mathematical morphological operation is to gradually increase the image boundary of region of foreground pixels whereas erosion mathematical morphological operation gradually shrinks the background [20]. Hence, it removes the noises.

The erosion of binary image f by a structuring element s is denoted by f θs which produces g=f θs. It reduces the size of regions of interest.

$$\text{If } g(x,\ y) = \begin{cases} 1, & 's'\ fits\ 'f' \\ 0, & otherwise \end{cases}$$

The dilation of an image f by a structuring element s is denoted by f ⊕ s which produces g = f ⊕ s. It adds a layer of pixels to both inner and outer boundaries of region.

$$\text{If } g(x,\ y) = \begin{cases} 1, & 's'\ hits\ 'f' \\ 0, & otherwise \end{cases}$$

The number of iteration for each erode operation is depends on how much noise presents in the video frame. So, here we applied dilation operation applied on the output of the erosion operation. These operations remove noise, isolation of individual elements and distinct element in an image and finding intensity holes in an image. Both operations take two pieces of data as input. One is the image which is to be dilated and another one is set of coordinate points known as structuring elements/kernel with the size of N × N. Each N × N size of frames is processed by different worker nodes of cloud environment using functional domain decomposition approach.

2.5 Blob Detection

It is a mathematical step to detect specific region based on some property in digital image. Blob made up of pixels having specific property in common and pixels which are combined together if they share common neighborhood pixels that have values greater than the threshold are the property of interest. After absolute differencing and noise removal applied in each video frames, the output of the pixel value is greater than binary threshold which is eligible to form a blob. All the adjacent neighbor of each such pixels are tested for eligibility and, if they satisfy, all those pixels are combined together to form blobs. In order to blob created by spurious movement, which are considerably in small, the size of detecting blob is limited by choosing blob threshold. By default, in this work the threshold value for blob detection is 15 and also applied smooth Gaussian filter with small kernel size (i.e., 3 × 3). Functionally decomposed kernels are processed by different worker node as individual thread. The output of this module is given as the feedback to detection system for learning.

2.6 Contour Detection and Tracking

Contour is the boundary of object pixel above the threshold value. It is edge based features which are insensitive to illumination change. Contour analysis output uses a linked list which consists of detected blobs. All the promising contours are found and reconstructed a full hierarchy of nested contour by using RETR_TREE Mode. The end points which are necessary for drawing the contour are extracted by using CHAIN_APPROX_SIMPLE method. In this work, once contour is identified, each contour is tracked separately with individual thread of different worker nodes in cloud environment. Fig. 4 shows the sample video frames and corresponding detect contour in the frame.

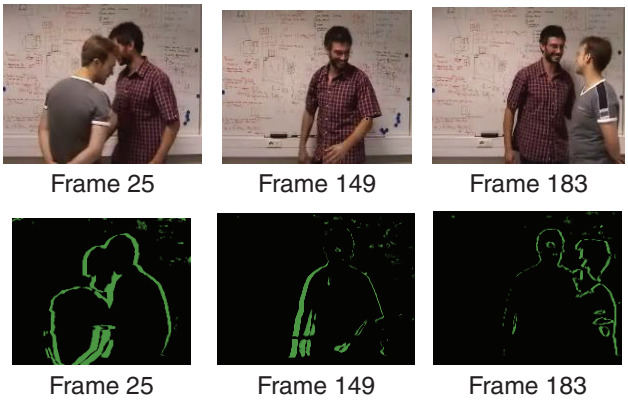


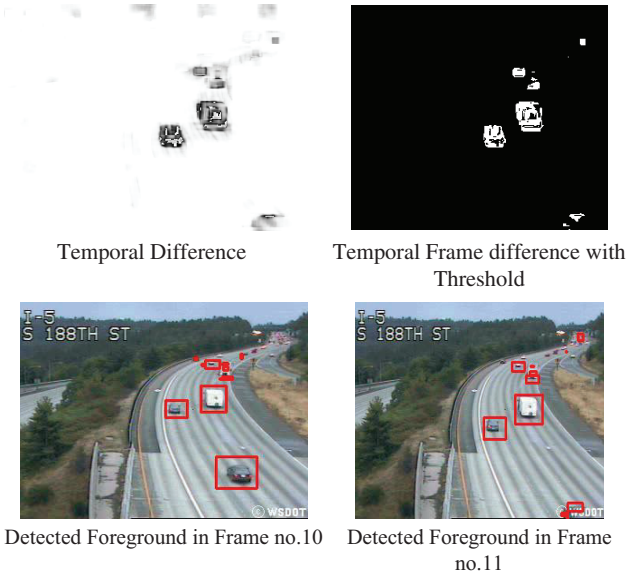
Figure 4: Sample capture video frame and its corresponding contour detected frames

The proposed algorithm is shown in the Fig. 5.

```
Input: Video
kernel of frame=n  #window size in frame
for each frames in video:
        #Each kernel of adjacent frame pixels are created as a
thread for parllelizing
for each thread:
for each row in frame kernel:
                        for each column in frame kernel:
#extract the moving object
                                Dk=|Ix,y,t-Ix,y,t-1|
                        Endfor
                Endfor
        #updates the reference image
        B_{t+1} = (1-α)B_t + αI_{x,y,t}
        #Apply erosion and dilation for noise removal and extracts the exact region of interests
        S= structuring element
        if g(x,y)= 's' fits in f(x,y):
                g(x,y)=1
        else:
                g(x,y)=0
        Detect blob of the object
        Apply Gaussian filter on blobs
        Detect the contour of  the objects
        Track the contour throughout the each frames
                Assign the tread to executor
        Endfor
Endfor
```

Figure 5: Pseudo code of proposed algorithm

The following Fig. 6 shows the temporal difference between contiguous frames and extracted foreground object using threshold, detected foreground objects.



Temporal Difference

Temporal Frame difference with Threshold

Detected Foreground in Frame no.10

Detected Foreground in Frame no.11

Figure 6: Temporal frame difference, with threshold output and detected foreground in contiguous frame

3 Performance Evaluation

We have run some tests in order to compare the performance of different execution. We set the different tests by varying the following parameters:

- Number of Video Frames
- Frame rate
- Video file size
- Video Resolution
- Kernel Dimension for processing

The performance of the proposed approach is evaluated in local standalone machine with Intel® Core™ i5-3380M CPU@ 2.90 Ghz with 4 GB RAM, GPU Node(NIVIDIA GeForce210 with 4 GB Ram) and Aneka Cloud composed of 3 nodes–1 manager and 2 executor where each arehaving the configuration of Intel Core i5 processor systems with 8 GB RAM. The entiremachines in the test were running in Windows 7 professional SP1 and .Net Framework 2.0 with EMGUCV. With the intention of checking the proposed approach, it is tested over several videos which are chosen from the CAVIAR and VISIOR data set. The accuracy of object detection is shown in Tab. 1 and also Tabs. 2–7 shows the total elapsed time of object detection and tracking in various video with varying kernel dimension for processing without multi-threading in standalone i5 processor, with multi-threading in standalone i5 processor, without multi-threaded GPU node, with multi-threaded GPU node, Aneka Cloud with single executor and Aneka Cloud with 2 Executor.

Table 1: Object detection accuracy of various video data sets with different resolution

Video data set	Detection accuracy (%)
Data set 1	90.7
Data set 2	93.5

(Continued)

Table 1 (continued)

Video data set	Detection accuracy (%)
Data set 3	94.8
Data set 4	97.6
Data set 5	98.5
Data set 6	98.2
Data set 7	96.8
Data set 8	98.2
Data set 9	98.5
Data set 10	96.3
Average	96.31

Table 2: Total elapsed time (in seconds) of object detection and tracking process without multi-threading in standalone i5 processor

Data sets	#Frames	Frame rate	File size	Resolution	3 × 3 window size	5 × 5 window size	7 × 7 window size	9 × 9 window size	11 × 11 window size
Data set 1	200	15	352 KB	160 × 130	726	449	423	350	320
Data set 2	158	15	121 KB	192 × 144	602	492	480	367	350
Data set 3	227	18	533 KB	200 × 164	756	470	450	388	335
Data set 4	53	10	124 KB	320 × 240	1161	629	581	422	409
Data set 5	599	30	5.15 MB	320 × 240	1310	1225	1020	850	730
Data set 6	887	10	18.5 MB	320 × 240	1400	1100	900	711	609
Data set 7	1700	60	3.46 MB	320 × 240	1500	1225	1020	850	730
Data set 8	4556	30	21.2 MB	320 × 240	3230	3100	2999	2800	2730
Data set 9	606	25	9.09 MB	352 × 288	1350	900	810	611	509
Data set 10	1496	25	15.3 MB	360 × 288	1475	1205	1000	820	715

Table 3: Total elapsed time (in seconds) of object detection and tracking process with multi-threading in standalone i5 processor

Video data sets	#Frames	Frame rate	File size	Resolution	3 × 3 window size	5 × 5 window size	7 × 7 window size	9 × 9 window size	11 × 11 window size
Data set 1	200	15	352 KB	160 × 130	378	350	320	315	295
Data set 2	158	15	121 KB	192 × 144	517	400	379	374	342
Data set 3	227	18	533 KB	200 × 164	400	370	350	330	320
Data set 4	53	10	124 KB	320 × 240	792	620	567	366	350
Data set 5	599	30	5.15 MB	320 × 240	970	987	900	660	600
Data set 6	887	10	18.5 MB	320 × 240	1050	887	799	545	499

(Continued)

Table 3 (continued)

Video data sets	#Frames	Frame rate	File size	Resolution	3 × 3 window size	5 × 5 window size	7 × 7 window size	9 × 9 window size	11 × 11 window size
Data set 7	1700	60	3.46 MB	320 × 240	1125	987	900	660	600
Data set 8	4556	30	21.2 MB	320 × 240	2125	1987	1900	1660	1600
Data set 9	606	25	9.09 MB	352 × 288	950	795	701	450	501
Data set 10	1496	25	15.3 MB	360 × 288	1105	957	870	640	589

Table 4: Total elapsed time (in seconds) of object detection and tracking process without multi-threading in GPU

Video data sets	#Frames	Frame rate	File size	Resolution	3 × 3 window size	5 × 5 window size	7 × 7 window size	9 × 9 window size	11 × 11 window size
Data set 1	200	15	352 KB	160 × 130	131	100	90	75	67
Data set 2	158	15	121 KB	192 × 144	209	136	97	82	78
Data set 3	227	18	533 KB	200 × 164	248	171	163	143	84
Data set 4	53	10	124 KB	320 × 240	261	228	103	88	81
Data set 5	599	30	5.15 MB	320 × 240	750	627	539	420	320
Data set 6	887	10	18.5 MB	320 × 240	650	500	400	320	215
Data set 7	1700	60	3.46 MB	320 × 240	750	627	539	420	320
Data set 8	4556	30	21.2 MB	320 × 240	1750	1627	1539	1420	1320
Data set 9	606	25	9.09 MB	352 × 288	550	450	325	220	115
Data set 10	1496	25	15.3 MB	360 × 288	725	597	501	400	295

Table 5: Total elapsed time (in seconds) of object detection and tracking process with multi-threading in GPU

Video data sets	#Frames	Frame rate	File size	Resolution	3 × 3 window size	5 × 5 window size	7 × 7 window size	9 × 9 window size	11 × 11 window size
Data set 1	200	15	352 KB	160 × 130	110	87	85	82	81
Data set 2	158	15	121 KB	192 × 144	115	91	75	70	62
Data set 3	227	18	533 KB	200 × 164	120	99	90	87	85
Data set 4	53	10	124 KB	320 × 240	122	87	75	71	69
Data set 5	599	30	5.15 MB	320 × 240	401	398	370	320	240
Data set 6	887	10	18.5 MB	320 × 240	277	257	220	200	115
Data set 7	1700	60	3.46 MB	320 × 240	401	398	370	320	240
Data set 8	4556	30	21.2 MB	320 × 240	1401	1398	1370	1320	1240
Data set 9	606	25	9.09 MB	352 × 288	178	147	120	100	90
Data set 10	1496	25	15.3 MB	360 × 288	387	367	340	299	215

Table 6: Total elapsed time (in seconds) of object detection and tracking process with Aneka Cloud-single executor

Video data sets	#Frames	Frame rate	File size	Resolution	3 × 3 window size	5 × 5 window size	7 × 7 window size	9 × 9 window size	11 × 11 window size
Data set 1	200	15	352 KB	160 × 130	90	67	62	57	50
Data set 2	158	15	121 KB	192 × 144	97	77	57	52	42
Data set 3	227	18	533 KB	200 × 164	90	77	77	65	58
Data set 4	53	10	124 KB	320 × 240	92	67	55	43	38
Data set 5	599	30	5.15 MB	320 × 240	380	345	330	287	205
Data set 6	887	10	18.5 MB	320 × 240	252	235	205	182	125
Data set 7	1700	60	3.46 MB	320 × 240	378	355	332	289	219
Data set 8	4556	30	21.2 MB	320 × 240	1390	1377	1325	1275	1228
Data set 9	606	25	9.09 MB	352 × 288	152	135	112	83	75
Data set 10	1496	25	15.3 MB	360 × 288	620	584	542	498	447

Table 7: Total elapsed time (in seconds) of object detection and tracking process with Aneka Cloud-2 executor

Video data sets	#Frames	Frame rate	File size	Resolution	3 × 3 window size	5 × 5 window size	7 × 7 window size	9 × 9 window size	11 × 11 window size
Data set 1	200	15	352 KB	160 × 130	67	45	37	30	25
Data set 2	158	15	121 KB	192 × 144	75	55	35	30	25
Data set 3	227	18	533 KB	200 × 164	75	52	45	40	35
Data set 4	53	10	124 KB	320 × 240	85	55	38	35	30
Data set 5	599	30	5.15 MB	320 × 240	350	325	297	252	267
Data set 6	887	10	18.5 MB	320 × 240	232	209	192	155	109
Data set 7	1700	60	3.46 MB	320 × 240	328	299	275	242	212
Data set 8	4556	30	21.2 MB	320 × 240	1375	1347	1295	1243	1192
Data set 9	606	25	9.09 MB	352 × 288	132	109	87	64	47
Data set 10	1496	25	15.3 MB	360 × 288	550	530	491	450	401

The data represented in the Tab. 1 show that object detection accuracy of various video data sets with different resolution which is calculated by the following formula.

$$a_r = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \times 100 \tag{3}$$

where T_p, T_n are the true positive and true negative rates. F_p, F_n are the false positive and false negative rates. From the aforementioned table, it is observed that we get high accuracy of object detection from high resolution video. But the processing time of high resolution video is high.

MSE is one of the measurements for identifying similarity between frames. If we are using MSE is only the measurement of identifying similarity between frames, it runs into problem. Hence in this work, we used Structural similarity index developed in [21].

$$\mathrm{SSIM}(x,\ y) = \frac{(2\mu_x\mu_y + c_1) + (2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4}$$

SSIM attempts to model the perceived change in the structural information of the image. It compares two window rather than compare entire image in MSE. SSIM value can vary between −1 to 1 where 1 indicates perfect similarity. The Fig. 7 shows the SSIM analysis between each frame. Here, SSIM is varied from 0.825 to 1. This results shown that, there is a change of luminance and/or contrast. But it will not the affect detection accuracy of foreground using temporal frame differencing algorithm.

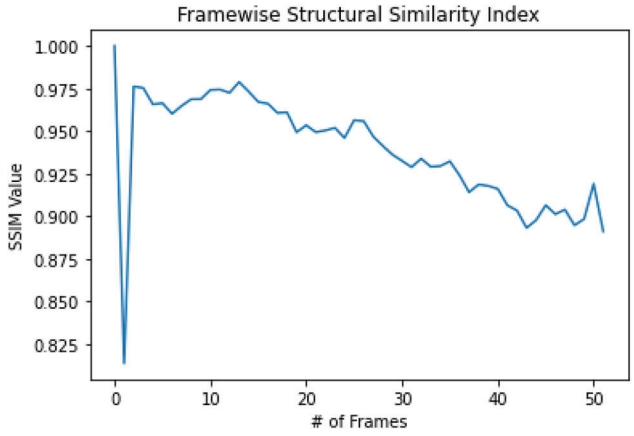


Figure 7: Frame wise SSIM analysis

The efficiency of the proposed approach is tested in terms of total elapsed time of object detection and tracking process and speedup ratio between Standalone system *vs.* GPU node *vs.* Aneka Cloud platform with and without multi-threading by varying kernel size of video frames are given in Tab. 2.

Fig. 8 shows histogram of processing time for object detection and tracking. It shown that object detection and tracking in private cloud environment (i.e., Aneka with 2 executor node) will outperform other parallel processing technique with hybrid decomposition technique.

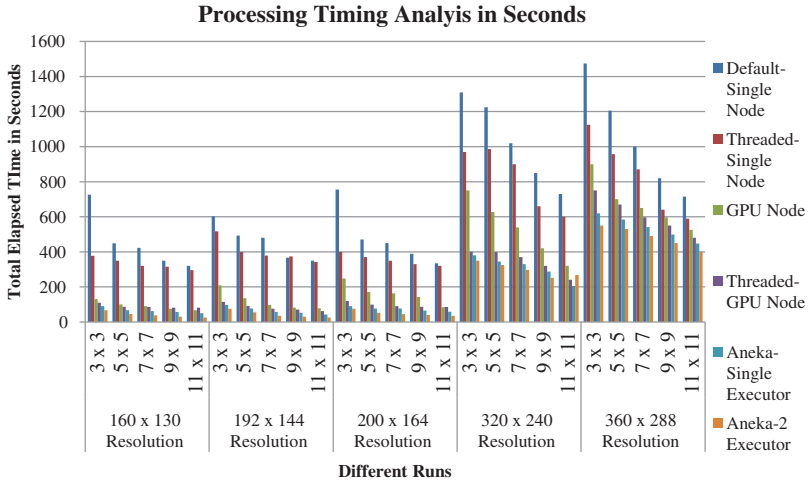


Figure 8: Processing time histogram of object detection and tracking

Figs. 9 and 10 shows the results of speedup ratio between standalone node *vs.* GPU node *vs.* Aneka Cloud single and 2 Executors with and without multi-threading. From the results, it is observed that Aneka cloud platform always gives better performance even though which is having communication overhead between nodes for detecting and tracking the objects in video without compromising detection accuracy and also without investing any specialized hardware platform.

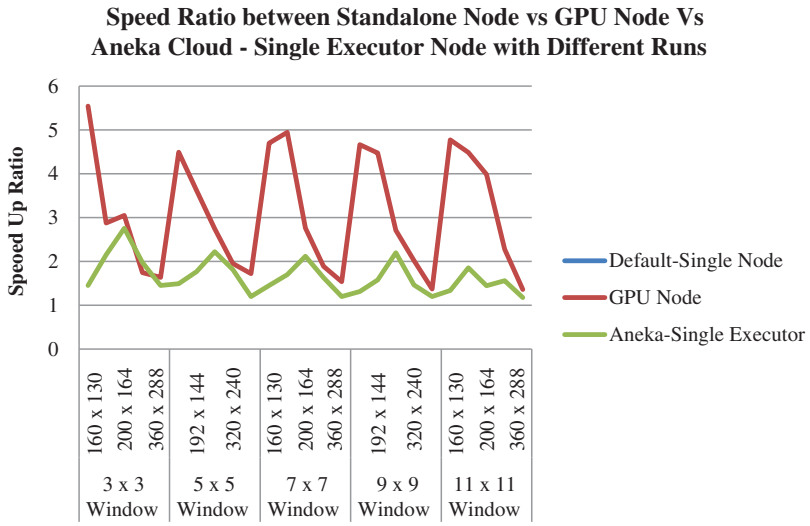


Figure 9: Speedup ratio between standalone node *vs.* GPU node *vs.* Aneka Cloud with single executor without multi-threading

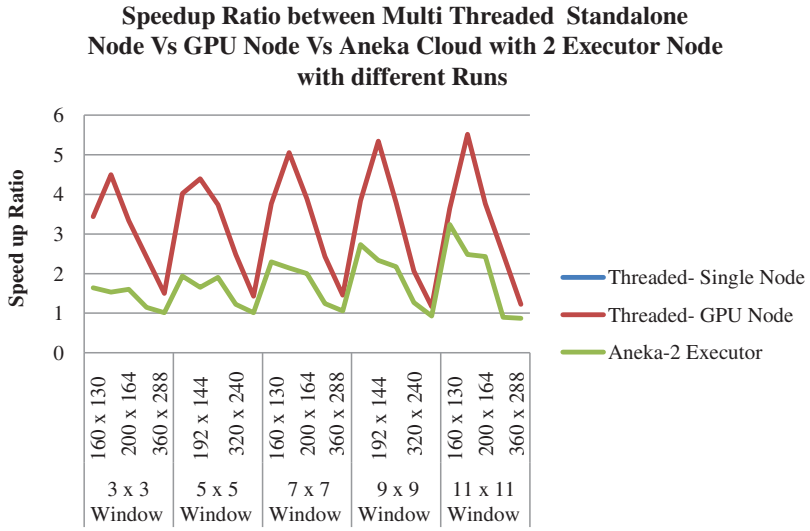


Figure 10: Speedup ratio between standalone node *vs.* GPU node *vs.* Aneka Cloud with 2 executor with multi-threading

4 Conclusion and Future Work

This work proposed a cloud computing based rapid object detection and tracking algorithm for video solution applied with temporal frame differencing and contour matching techniques using functional

decomposition technique. In this work, each functional component of temporal object detection and tracking algorithms are executed parallelly by the worker node. Again, these functional decomposed components are internally parallelized using inherent sequential parallelizing mechanism which is executed by the individual workers nodes of Aneka cloud environment. The performance of proposed approach is compared and evaluated with standalone machine, GPU node and Aneka Cloud environment. The performance of cloud computing based object detection and tracking outperforms well compare to other platform without investing any specialized hardware architecture. In future, this work is plan to extend by extracting the various features (like LBP, HOG) of detected blobs and improve the accuracy of detection and tracking of blobs and improve the computation as a service for very swift object detection and tracking algorithm through the design of application level task scheduler for cloud environment. This work more suitable for automation field. We plan to extend this work in real time both single and multiple camera whereas currently we used to test on only stored videos. Also, design the storage cloud architecture for video processing and surveillance application.

Acknowledgement: I am extremely thankful to Management, Principal, Head of the department of Computer Science and Engineering, Ramco Institute of Technology, Rajaplayam for providing me laboratory facility in RIT-AI Research Centre to carry out this work in successful way.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Czyzewski, G. Szwoch, P. Dalka, P. Szczuko, A. Ciarkowski *et al.,* "Multi-stage video analysis framework. Video surveillance," in *Intech Open Book Series*, Intech Open Ltd., United Kindgdom, vol. 9, pp. 147–172, 2011.

[2] H. S. Parekh, D. G. Thakore and U. K. Jaliya, "A survey on object detection and tracking methods," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 2, pp. 2970–2979, 2014.

[3] M. Gomathynayagam and K. Ramar, "A survey on real time object detection and tracking algorithms," *International Journal of Applied Engineering Research*, vol. 10, no. 5, pp. 8290–8297, 2015.

[4] M. Álvarez and M. Lopez, "Road detection based on illuminant invariance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 184–193, 2011.

[5] P. Spangnolo, T. D. Orazio, M. Leo and A. Distante, "Moving object segmentation by background subtraction and temporal analysis," *Image and Vision Computing*, vol. 24, no. 5, pp. 411–423, 2006.

[6] W. Zhang, P. Xu, L. Duan, W. Gong, X. Liu *et al.,* "Towards a high speed video cloud based on batch processing integrated with fast processing," in *Proc. Int. Conf. of Identification, Information and Knowledge in the Internet of Things (IIKI)*, Beijing, China, pp. 28–33, 2014.

[7] D. Melpignano, L. Benini, E. Flamand, B. Jego, T. Lepley *et al.,* "Platform 2012, a many-core computing accelerator for embedded socs: Performance evaluation of visual analytics applications," in *Proc. 49th Annual Design Automation Conf.*, San Francisco, California, pp. 1137, 2012.

[8] S. Solehah, S. Nizamyaakob, Z. Kadim and H. HockWoon,"Extraction of moving object using frame differencing, ghost and shadow removal," in *Proc. 5th IEEE Int. Conf. on Intelligent Systems, Modeling and Simulation*, Malaysia, pp. 229–234, 2014.

[9] J. Parsola, D. Gangodkar and A. Mittal, "Efficient storage and processing of video data for moving object detection using hadoop/map reduce," in *Proc. Int. Conf. on Signal, Networks, Computing and Systems*, Springer Lecture Notes in Electrical Engineering, India, vol. 395, pp. 137–147, 2017.

[10] S. K. Maharana, P. Ganeshprabhakar and A. Bhati, "A study of computing for retinal image processing through matlab," *International Journal of Cloud Applications and Computing*, vol. 2, no. 2, pp. 59–69, 2012.

[11] M. Gomathynayagam and K. Ramar, "Reliable object recognition system for cloud video data based on ldp feature," *Elsevier Journal of Computer Communications*, vol. 149, pp. 343–349, 2020.

[12] K. A. Joshi and D. G. Thakore, "A survey on moving object detection and tracking in video surveillance system," *International Journal of Soft Computing and Engineering (IJSCE)*, ISSN, vol. 2, no. 3, pp. 2231–2307, 2012.

[13] M. Zhu, S. Shuifa, S. Han and H. Shen, "Comparison of moving object detection algorithms," *World Automation Congress (WAC)*, Mexico, vol. 1, pp. 35–38, 2012.

[14] R. Radke, S. Andra, O. Al-Kofahi and B. Roysam, "Image change detection algorithms: A systematic survey," *IEEE Trans. Image Processing*, vol. 14, pp. 294–307, 2005.

[15] S. Shih-Wei, F. Yu-Chiang, F. Huang and H. Mark Liao, "Moving foreground object detection via robust sift trajectories," *Journal of Visual Communication Image*, vol. 24, pp. 232–243, 2012.

[16] L. Dawei, X. Lihong and D. Goodman, "Illumination-robust foreground detection in a video surveillance system," *IEEE Trans. on Circuits and System for Video Technology*, vol. 23, no. 10, pp. 1637–1650, 2013.

[17] M. Gomathynayagam and K. Ramar, "A design of hybrid workflow model for real time object detection using temporal frame differencing algorithm: A cloud computing approach," *Asian Journal of Research in Social Science and Humanities*, vol. 6, no. 10, pp. 2099–2113, 2016.

[18] C. Vecchiola, K. Nadiminti and R. Buyya, "Image filtering on. Net-based desktop grid," in. *Proc. 6th Int. Conf. on Grid and Cooperative Computing (GCC 2007, IEEE CS Press, Los Alamitos, CA, USA)*, Urumchi, Xinjiang, China, pp. 16–18, 2007.

[19] R. Cucchiara, C. Grana, A. Prati and R. Vezzani, "Probabilistic posture classification for human-behavior analysis," *IEEE Transactions on Systems*, vol. 35, no. 1, pp. 42–54, 2005.

[20] J. Heikkila and O. Silven, "A real-time system for monitoring of cyclists and pedestrians," in *Proc. Second IEEE Workshop on Visual Surveillance*, USA, pp. 74–81, 1999.

[21] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transaction on Image Processing*, vol. 13, no. 4, pp. 1–14, 2004.