Comparison Of Processing Time Of Different Size Of Images And Video Resolutions For Object Detection Using Fuzzy Inference System

Yogesh Yadav, Rajas Walavalkar, Sagar Suchak, Abhishek Yedurkar, Swapnil Gharat

Abstract: Object Detection with small computation cost and processing time is a necessity in diverse domains such as :traffic analysis, security cameras, video surveillance etc .With current advances in technology and decrease in prices of image sensors and video cameras , the resolution of captured images is more than 1MP and has higher frame rates. This implies a considerable data size that needs to be processed in a very short period of time when real-time operations and data processing is needed. Real time video processing with high performance can be achieved with GPU technology. The aim of this study is to evaluate the influence of different image and video resolutions on the processing time, number of objects detections and accuracy of the detected object. MOG2 algorithm is used for processing video input data with GPU module. Fuzzy interference system is used to evaluate the accuracy of number of detected object and to show the difference between CPU and GPU computing methods.

Keywords: Image Processing, Fuzzy Interference System, Image Resolution, MOG2(Measure of Gaussians), BSM (Background Subtraction Method),

1 INTRODUCTION

The recent advances in low-cost imaging solutions and increasing storage capacities, there is an increased demand for better image quality in a wide variety of applications involving both image and video processing. While it is preferable to acquire image data at a higher resolution, a wide range of scenarios where it is technically not feasible. In some cases, it is the limitation of the sensor due to low-power requirements as in satellite imaging, remote sensing, and surveillance imaging. In other cases, it is the limitation of the sensed environment itself, for e.g. the presence of atmospheric clutter, background noise and unfavorable weather. In some cases, it is a combination of both, for e.g. the acquisition of medical images is limited both by the physical issues of MRI imaging, as well as the time constraints of subjecting the magnetic field to patients without becoming a health hazard. Thus, to acquire image data from the input video at optimum resolution from the sensors so that it can be processed within a short interval for real-time application is a necessity[1]. For example: Consider intelligent traffic street lighting system where intensity of the street lamps depend on the traffic of the road. If there is no traffic on the road (i.e. no presence of vehicle), the lamp will glow with threshold intensity and for roads with maximum traffic, it will glow with highest intensity[2]. Thus, intensity of street lamps depend on the traffic of the road. The traffic video will be continuously captured by the camera sensor. From the video, image frames will be selected within some time interval and then frames will be transferred to the processor where it will be processed and the corresponding intensity value will be evaluated and send to the street lamp to glow with that intensity.

- *Yogesh Yadav, Rajas Walavalkar, Sagar Suchak, Abhishek Yedurkar, Student, Computer engineering Dept. , Rajiv Gandhi Institute Of Technology, Mumbai University . Email – yogeshoyadav08@gmail.com*
- *Swapnil Gharat, Project Guide, Faculty in Computer Engineering Dept, Rajiv Gandhi Institute Of Technology.*

The whole process will take place in a quick span of time, thus the selection of camera sensors is of utmost importance. The resolution of the image captured by the sensor, video quality and the processing time for evaluating number of objects are the factors to be considered for the selection of camera sensors[2]. If image resolution and the video quality is low (2MP), the processing time will be faster but the accuracy to detect the number of objects will be low. Similarly, for high video quality and image resolution (8MP), accuracy to detect number of objects will be high but processing time will be too long[2]. Thus, a right balance need to be present between processing time and image resolution for the captured image so that it can detect number of objects with optimum processing time. With the proliferation of many-core platforms, multi- and many-thread implementations gain in importance, use of GPU technology for real time video processing will certainly meet the required demands[3]. Graphics Processing Units (GPUs) have evolved to support general purpose applications from different market domains, suitable for data-intensive applications with high-throughput requirements and SIMD style parallelism [3]. Consequently, GPUs have also been considered for high-performance image and vision computing. Since MoG is an parallel application—individual pixel are processed independently—it is a suitable candidate for GPU mapping for video input to map the traffic density continuously after some specified time interval[6]. Thus, the study involves comparison of processing time for different image and video resolutions for CPU and GPU computing modules[4]. The comparison of different image and video resolution is done with the help of fuzzy interference system[4].

2 RELATED WORK

With GPUs evolving to support general-purpose applications, researches have started for image processing with GPU platforms to improve throughput and efficiency[4]. The technique presented in this paper focuses on GPU optimizations with Background Subtraction algorithms (BS)[1]. MOG- BS algorithms is used for real time video processing The parallel threads in a GPU execute in lock-step mode. All threads perform the same amount of computation even with variable number of Gaussian

components. In result,the thread with the most Gaussian components determines the latency of all parallel threads. This leads to speed up over serial implementation.

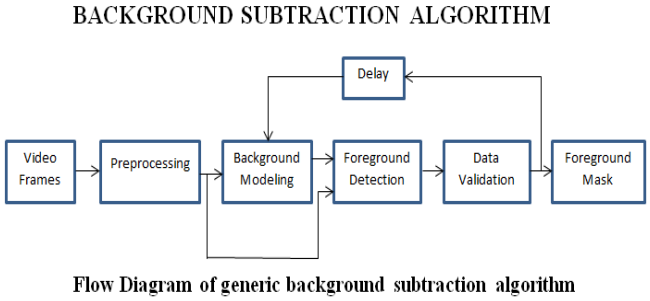

Fig 1: BSM

3 MOG IMPLEMENTATION

The input is a video frame and Gaussian parameters. The algorithm loops through all pixels in the frame. For each pixel, the algorithm first classifies the pixel's Gaussian components into match or non-match components. A component matches if the component's mean is within a certain range of the current pixel value. Gaussian parameters are updated based on match classification[1] .In case that no match exists, algorithm creates a new Gaussian component, called virtual component, and replaces the virtual component with the Gaussian component with smallest weight value. Then, the components are ranked based on their weight over standard deviation ratio and sorted by rank . Starting from the highest rank component, the algorithm declares a pixel to be background based on rank and closeness in matching with the current value. When finding the first match, the comparison stops and the algorithm continues with the next pixel. To realize real-time processing, acceleration is necessary. MOG loop iterations over pixels are independent, thus each pixel could be operated on in parallel. Hence, MoG is embarrassingly parallel. A GPU with the massively parallel compute units is a suitable target platform.

3.1 Algorithm: MOG Pseudo-code

1: function MOG (in Frame, in out Gaussian, out Foreground)
2: for i = 0 to numPixel do
3: for k = 0 to numGau do
4: diff[k] = abs(mk - pixel)
5: if diff[k] < □1 then
6: update wk, mk and sdk
7: match = 1
8: else
9: update w
10: end if
11: end for
12: if !match then
13: Create virtual component
14: Find the smallest w
15: end if
16: for all gaussian do
17: Calc Rank for gaussian
18: end for
19: for all gaussian do
20: Sort Component by Rank
21: end for
22: Foreground = 1
23: for k = HighestRank to 0 do
24: if k>=#2 && diff[k]=sdk < #1 then
25: Foreground = 0
26: break27: end if
28: end for
29: end for
30: end function

4 FUZZY INTERFERENCE SYSTEM

Fuzzy inference systems are also known as fuzzy-rule-based systems, fuzzy models, fuzzy associative memories (FAM), or fuzzy controllers when used as controllers. Basically a fuzzy inference system is composed of five functional blocks[4]:

- a rule base containing a number of fuzzy if-then rules
- a database which defines the membership functions of fuzzy sets used in the fuzzy rules
- a decision-making unit which performs the inference operation on the rules
- a fuzzification interface which transforms the crisp inputs into degree of match with linguistic values
- a defuzzification interface which transform the fuzzy results of interference into a crisp output

5 MEASURING FIS PERFORMANCE FOR IMAGE AND VIDEO RESOLUTIONS

Input parameter:

- Processing time(PT)
- Image size(IS)

Output parameter:

- Number of Objects (Nob)

Descriptors for Input parameter

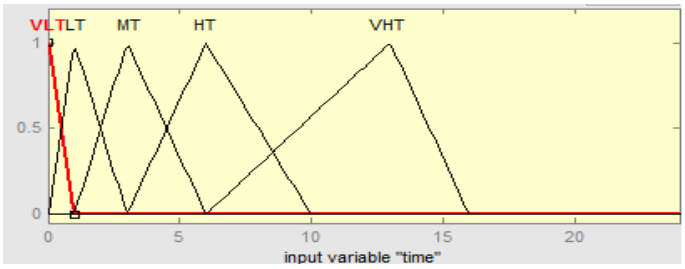

***Fig 2:** Membership function of Processing Time (T)*

- PT:VLT,LT,MT,HT,VHT
  - VLT: Very Low Time
  - LT: Low Time
  - MT: Medium Time
  - HT: High Time
  - VHT: Very High Time

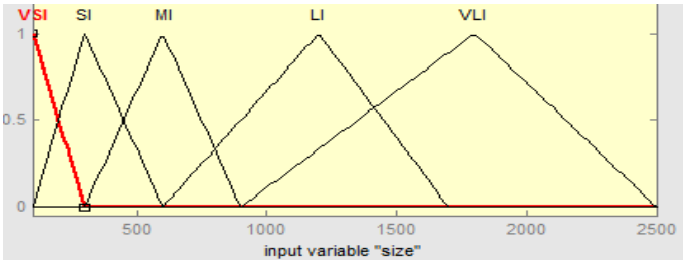

Fig 3: *Membership function of Image Size(IS)*

- IS:VLS,LS,MS,HS,VHS
  - VSI: Very Small Image
  - SI: Small Image
  - MI: Medium Image
  - LI: Large Image
  - VLI: Very Large Image

Descriptors for Output parameter

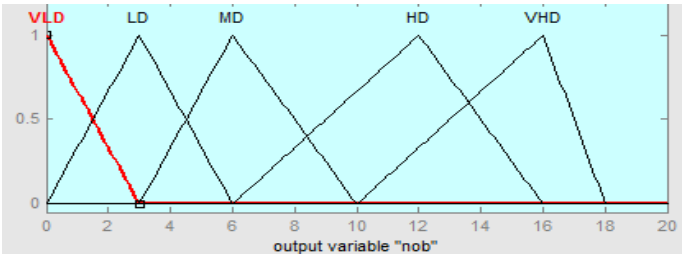

Fig 4: *Membership function of NOB*

- NOB: VLD,LD,MD,HD,VHD
  - VLD: Very Low Detection
  - LD: Low Detection
  - MD: Medium Detection
  - HD: High Detection
  - VHD: Very High Detection

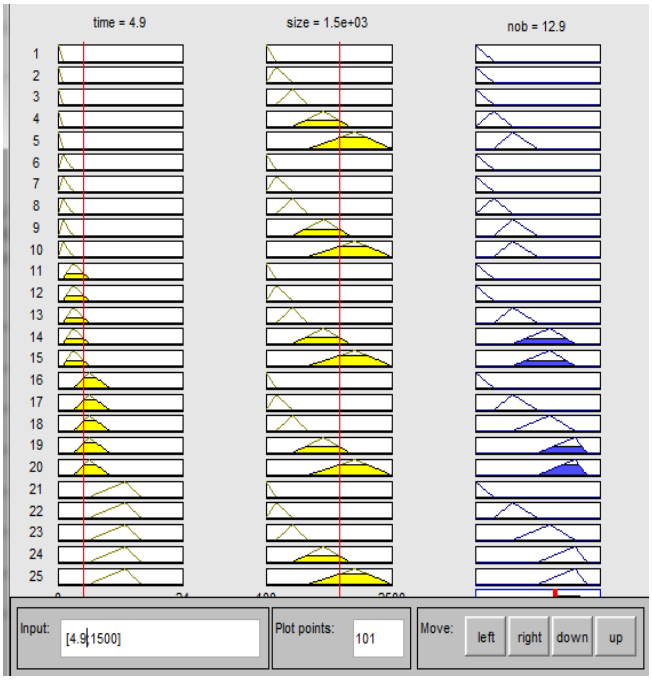

Fig. 5: *Estimation of Accuracy (Nob) for image size of 1500x1500 pixels*

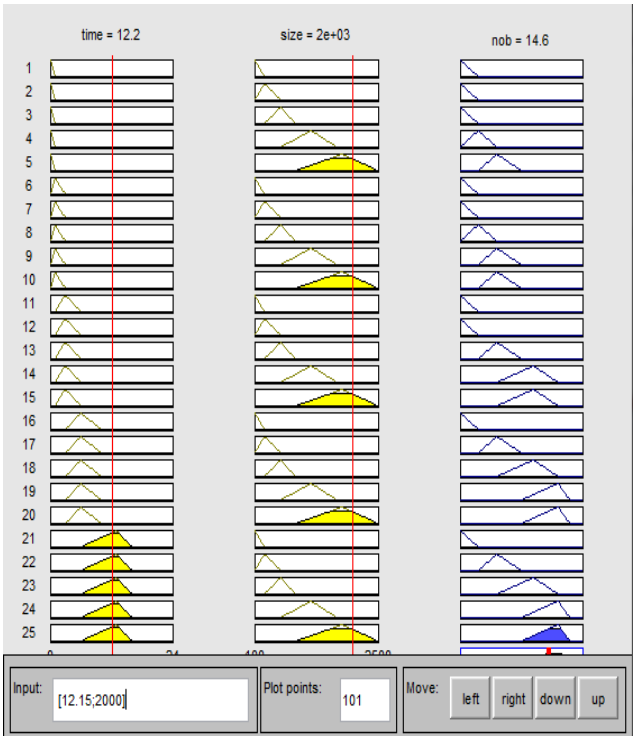

Fig. 6: *Estimation of Accuracy (Nob) for image size of 2000×2000 pixels*

Surface diagram

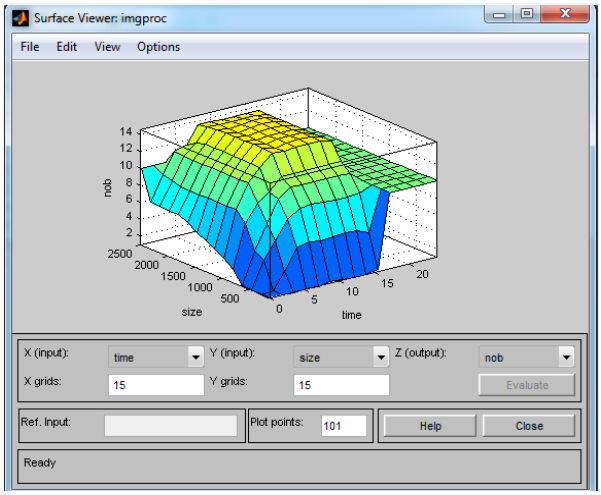

Fig 7:*3D Surface View Diagram*

A three-dimensional curve that represents the mapping from image size(x axis) and processing time(y axis) to accuracy(z axis). As this curve represents a two-input one-output case, you can see the entire mapping in one plot.

6 Video Processing With GPU Module:

In video monitoring, background subtraction method is an important technique to firstly model the background and then to detect the moving objects in the scene captured by a static camera[6]. In crowded urban, traffic congestion is a severe challenge to any current background modeling algorithm. In this context, slowly moving vehicles or the pedestrians with high density can make the scene chaotic and the background ruined. Furthermore, the impact from

outdoor illumination also affects the background result. Generally speaking, the input frames suffering from those effects are called disordered frames (or images) and must be eliminated. To deal with this problem, frame difference is known as the first strategy which subtracts the background from each input frame; the background, maintained during the process, represents the stable scene after removing all non-stationary elements. Mixture of Gaussian (MOG) is the method used for processing the frames of video. When a single Gaussian density function is not capable enough to deal with dynamic scenes, MOG allows to model several features for each pixel and also solves the challenge of environmental variation. Thus techniques such as frame differencing, run quickly on CPU platforms, but their accuracy is not sufficient for most computer vision problems. In contrast, adaptive parametric background modeling techniques, such as the state of the art Mixture of Gaussians (MoG2) are more robust and speed up the computation.

7 IMPLEMENTATION:

In our study, we have implemented background subtraction technique MOG2 using GPU module of Raspberry Pi 3 to detect object in the input video data that is feed to the processor from the camera sensor[6]. The camera sensor continuously accepts video data from the environment and transfers it to the processor. Here, the processor uses the gpu module to ensure that video data is processed within fraction of seconds and to lower down the load on the processor as memory is continuously being upgraded with new input video data[6]. The MOG2 algorithm is implemented with Open CV software[7] .

MOG2 program code:

```
import numpy as np
import cv2
import timeit

cap = cv2.VideoCapture('people-walking.mp4')
fgbg = cv2.createBackgroundSubtractorMOG2()
i=0
proc=0
while(i<200):
    start = timeit.default_timer()
    ret, frame = cap.read()
    fgmask = fgbg.apply(frame)
    i=i+1
    stop = timeit.default_timer()

    cv2.imshow('fgmask',frame)
    cv2.imshow('frame',fgmask)
    cd=stop – start
    proc= proc + cd
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
avg = proc / I
print ‘ Avg Time’
print avg

cap.release()
cv2.destroyAllWindows()
```

The mog2 program for background subtraction takes video data as input. It then captures frame from the videos within specified time interval. It calls the MOG GPU Function and subtracts the frame with the background to get the object in the foreground image. It generates an image as a result, but the image is generated for all the frames continuously and thus the output displayed is also a Video with only showing objects in the foreground video. Thus real time video processing can be achieved with MOG2 algorithm[1].

Fig 8: *Result for MOG2 Algorithm*

8 CONCLUSIONS

This paper discussed and evaluated various technique for different image and video resolution for object detection for CPU as well as GPU technology. Thus if considering only CPU technology for video processing , as video size increases , the accuracy to detect number of objects increases but the processing time also increases. Therefore, a right balance need to be present during selecting the camera sensor so that it can process the video efficiently with optimum accuracy.

Sr no	Video Size (KB,MB)	Average Procesing time(sec)	Video name (mp4)
1	641.9KB	0.0398825	Cars
2	8.30MB	0.116756	People-walking
3	274.5MB	0.4457155	FroggerHighway

Table 1: *Processing time for different video sizes for GPU*

Video Processing with GPU technology for object detection, as video size increases the processing time[4] also increases but with very small range i.e even if the video size is increased to a greater extent (say 1080p)[6], the processing time for GPU will increase but within the range of seconds/milli seconds. Thus real time video processing for object detection with GPU computing modules is practically feasible with results that can provide optimum performance[6].

9 REFERENCES

[1] Chulian Zhang, Hamed Tabkhi and Gunar Schirner, “A GPU-based Algorithm-specific Optimization for High-performance Background Subtraction “.

[2] Prem Kumar.V, Barath.V, Prashanth.K, “Object counting and density calculation using matlab”.

[3] Antonios Georgantzoglou, Joakim da Silva and Rajesh Jena, “Image Processing with MATLAB and GPU”.

[4] Deepali Shinde, Mithilesh Said, PratiknShetty, Swapnil Gharat, “Optimizing real time GPU kernels using Fuzzy Inference System” IJARSE, Vol. No. 2, Issue No.9 , ISSN-2319-8354(E).

[5] Jingfei Kong,Martin Dimitrov, Yi Yang, Janaka Liyanage, Lin Cao, Jacob Staples, Mike Mantor, Huiyang Zhou, “Acclerating MATLAB Image Processing Toolbox Functions on GPUs”.

[6] V. Kastrinaki, M. Zervakis, K. Kalaitzakis, “A survey of Video processing techniques for traffic applications”, Image and Vision Computing 21(2003) 359 – 381.

[7] https://pythonprogramming.net/mog-background-reduction-python-opencv-tutorial.html

[8] http://in.mathworks.com/help/fuzzy/newfis.html

[9] http://opencv24pythontutorials.readthedocs.io/en/stable.htm