Static Logo Detection and Implementing its Methods on GPUs

Alireza Keshavarzian,
Electrical Engineering Department
Amirkabir University of Tehran
Tehran, Iran.
Email: Alireza.keshavarzian@aut.ac.ir

***Abstract*—I implement an algorithms for logo detection on GPUs. I apply plentitude of methods to reduce the amount of computation to gain the efficient performance. All of the functions such as Sobel, gradient, Histogram, etc. are implemented on GPUs to enhance the speed of the program. In this project, I use zero-copy memory in lieu of global memory to eliminate the latency of transferring the data among devices and host and enhance the PCIe transfer rate. Moreover, I neglect the inner portion of the videos' pixels which rarely consists of logos to boost the speed and consumption. To put it in a nutshell, I manage to decrease the time taken by a factor of 9.19 compared to the implementation of this method on host. The algorithm is generalized for other kind of detection and can be used in inpainting of portions of the picture.**

***Keywords*—Parallel, GPU, image processing, Logo, detection**

I. INTRODUCTION

LOGOS whether dynamic or static have occupied a majority of TV programs shown in all over the world. Germane to this fact, not only are these logos are representative of name of the videos or its ownership but could also cease the plagiarism and illegal copying of videos which increase the chance of producer's right to remain protected.

Generally, logos in videos are divided into two categories which I will aptly explain them in follows.

1- Opaque logo: logos in which all data placed behind the logos is covered completely in such a way that all information behind them are totally damaged.
2- Semi-transparent logo: unlike the former type, semi-transparent does not spoil data behind them entirely. Based upon the degree of the transparency, part of the videos which placed behind this kind of logos could be imperfectly visible and could be detectable.

It should be also pointed out that both opaque and semi-transparent logos can be either animated or statics. In this project the animated logos are not considered as an issue to solve [1]. This project has been tested among numerate videos having static logos. There are no constraints for this algorithm to detect logos correctly.

One potential application of this project is to allow elimination of logos during live streaming. The purpose for such an application is required as some news footage are only streamed from a certain international channel and other news channels are to replay the exact video with the logo of the original channel. Some countries have political conflicts in a way that they do not want the people of their country to be aware of the original channel as it is regarded as political propaganda be played in the news channels all around the world. Although Logo detection has innumerable benefits, in artificial intelligence, they play an important role.

Logo detection algorithms are mostly applied on gradient average of different images on logo's region in such a way that its result converge to constant value with the passage of time; while, the gradient in other portion of images alters differently based upon its pixels. Hence, the average of the gradient of the region which is located of the logos has higher value. Although another solutions are still available for this controversial problem, adopting this method culminate in reliable results without any limitation.

In this report, we design a specific filter to galvanize or weaken the gradient of each frames based on the existence of logo leading to detecting the logo in fewer frames. Finally, the parallel equivalent of algorithm and independency among the operations culminate in high level of available parallelism. Therefore, I implement these methods and algorithms which are by far suitable for parallel architecture on GPUs to gain the preferable results. In other words, the high execution speed offered by parallelization is more than meets the eyes.

II. PROPOSED METHOD AND BACKGROUND

The first step to start the image processing and detecting the logos is applying some kind of filters which ease the rest of operations of logo detection. These filters consist of $boxfilter$ and $sobel$ to create an image in which the edge of frames are appeared. *Boxfilter* mainly is used to make the frames of videos blur. Actually, this operation paves the way for *sobel* filter to

make clear the edges in the frames.

Box filter or mean filter is a linear filter in which each pixel in the frame has a value equal to the mean of its neighborhood pixel [1]. It is a form of low pass filter and mostly is written as 5×5 matrix, indicated below.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

After all, by dividing the magnitude, resultant by the multiplication of matrixes, the value of specific pixel placed in the center of its neighborhood is calculated.

Another step toward preparing the ground to detect the logos is finding edges and excluding extra information from images. Sobel filter pave the way for achieving this objective. This algorithm creates an image emphasizing edges. The operator uses two 3×3 matrixes which are convolved with the original frames to calculate the approximated value of derivative one for horizontal and vertical changes. If we assume A is the original frame, G_x and G_y are the matrixes which contain the horizontal and vertical derivative approximations [3]. The computations are as follows:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \quad (1)$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A \quad (2)$$

At each point in image, the resulting gradient approximation can be merged to calculate the gradient magnitude:

$$G = \sqrt{G_x^2 + G_y^2} \quad (3)$$

The chief objective of videos and movies are delivering the contention of news, documentary or some stuff like that to the audience. Hence, it is so important not to deliberately corrupt the videos by placing the logos on inappropriate places. Be that as it may, the location of logos mostly placed on the corner of the frames in such a way that the damages caused by logo become negligible. This is a well proven fact that the focus of cameras capturing the videos are mostly on the center of the frames. To put it in other words, the chance of existence of logos in the center of videos is around zero. To delineate, by going further to the corner of the images the chance of existence of logos increases. Hence, adopting an appropriate kind of filter could reinforce the algorithm and reduce the amount of energy consumed during the processes. The filter which meet this demand for us is similar $sinc$ function which is showed below [4] :

$$x_3 = -\gamma \frac{\sin\left(\pi\sqrt{\alpha_1^2 x_1^2 + \alpha_2^2 x_2^2}\right)}{\pi\sqrt{\alpha_1^2 x_1^2 + \alpha_2^2 x_2^2}} + \beta \quad (4)$$

For applying this filter on image, first of all we should compute the value of a_1 and a_2. As Figure 1 demonstrates the highest point in x_1 dimension is expected to be the tangent of the first and the last row of image. Likewise, the highest point in x_2 dimension is expected to be tangent to the first and last column of image. Hence the value of a_1 can be computed by finding the second extremum of the function in $x_1$and x_2 directions. Assume that the size of frame of the video is $M \times N$. In order to match perfectly on the image formula which is demonstrated bellow should be satisfied.

$$f_{x_1}'\left(\frac{M}{2}\right) = 0 \quad (5)$$

$$f_{x_2}'\left(\frac{N}{2}\right) = 0 \quad (6)$$

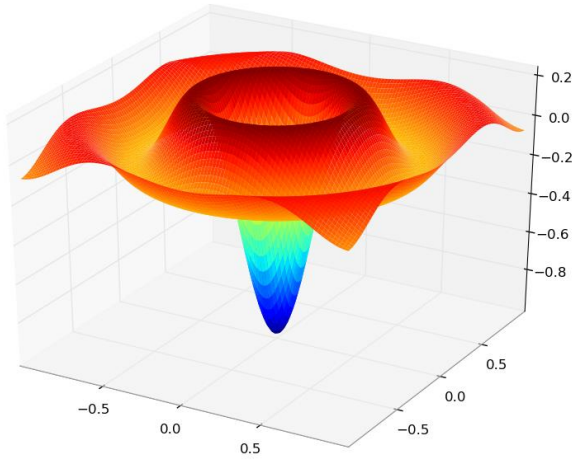

Figure 1: Sinc filter which is proposed as an appropriate filter

The value of f' is zero in plentitude of points. However, as the Figure 1 shows we need to find the second extremum. By solving the equation of (5) and (6) we could solve our equation. In this point the value of f' is zero. For simplicity, we assume both dimensions are equal. Hence the $a_1 = a_2$ and (7) is derived. As I elucidate above, I consider $M = N$ for the simplicity of equations.

$$\tan\left(\alpha\pi\sqrt{\frac{M^2}{2}}\right) = \alpha\pi\sqrt{\frac{M^2}{2}} \quad (7)$$

Thus, α is calculable as shown in (8), α must be computed for the second extremum.

$$\alpha = \tan^{-1}\frac{\alpha\pi\sqrt{\frac{M^2}{2}}}{\pi\sqrt{\frac{M^2}{2}}} + 2k\pi \quad (8)$$

By obtaining α using (8), the minimum and maximum values of (4) are shown in

$$f_{min} = -\gamma + \beta \quad (9)$$

$$f_{max} = -\frac{\gamma \sin\left(\alpha\pi\sqrt{\frac{M^2}{2}}\right)}{\alpha\pi\sqrt{\frac{M^2}{2}}} + \beta \quad (10)$$

The filter is expected to reinforce edges in the area in which the likelihood of existence of logo is more and weaken the edges in the regions in which the likelihood of existence of the logo is less. The desired fortification and attenuation of edges in different regions of videos vary depending on the video content and logo itself. These values can be adjusted by setting the values of maximum and minimum using (9) and (10), respectively, by changing γ and β.

Next step to cope with the issue is calculating the gradient of the frames after applying the sinc filter on our images. The gradient of images is given by the formula:

$$\nabla f = \frac{\partial f}{\partial x}\hat{x} + \frac{\partial f}{\partial y}\hat{y} \qquad (11)$$

Where $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ are the gradient in x and y respectively. However the gradient direction can be calculated by:

$$\theta = \tan^{-1}\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right) \qquad (12)$$

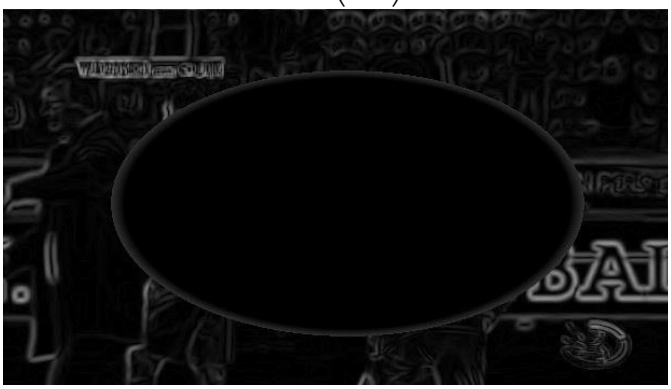

Figure 2: frame in which the sinc function applied

By what following, our last objective due to equation (4) is obtaining the β and γ. To detect the logo properly we should eliminate the extra information from images. By applying the gradient operation on frame, we could deal with this issue. I take at most 500 frames to recognize the logos in videos properly. However, if the difference between gradient become constant after a while, I stop computing gradient on the rest of the frames. By doing so, as the number of frames to detect logos decreases, so does the amount of energy. Due to (4) the equation (13) is derived. In this equation

$$E_q = f(x_1, x_2).(|\nabla l_l - \nabla l_{l+q}|) \qquad (13)$$

After that all extra information should be eliminated from the gradient of the images as shown in (14). In this equation Q is the number of frames which I've planned to calculate the gradient of them:

$$I_{l-clear} = \sum_{q=1}^{Q} |f(x_1, x_2) \times I_l - E_q| \qquad (14)$$

At the end of this procedure, all of the gradient frames which is denoted as $I_{l-clear}$ should be added with each other and then divide them to the number of frame which has been considered as sample. Delving further to the issue we derive equation (15). By dividing the E to the number of the frames, I achieve the proper result that could be suitable for our logo detection.

$$E = \sum_{i=-\theta}^{\theta} \sum_{j=-\theta}^{\theta} I_{l-clear}(i,j) \qquad (15)$$

In this condition, it is supposed that the logos in the videos just appear without any other information. Our assumption about the last picture is black screen in which white logos placed on.

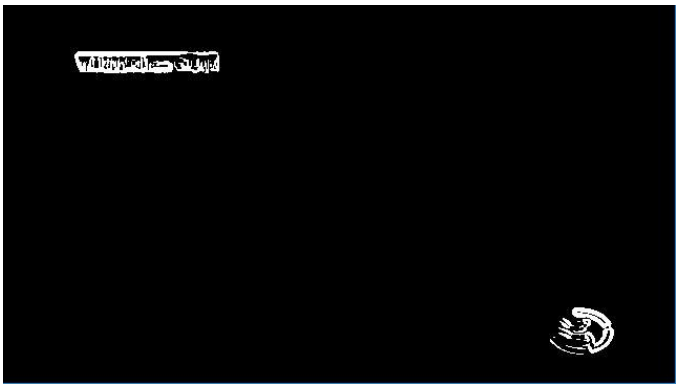

Figure 3: last frame on which gradient function is applied

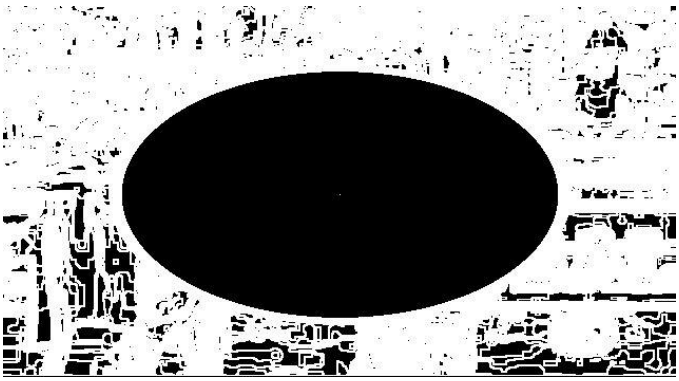

Figure 4: first frame on which gradient function is applied without getting any average

Finding appropriate number of frames to detect logos properly is another controversial issues. There are some methods to inspect logos in frames, I will explain them bellow, adopt one of them and delineate it.

1- After applying the sobel filter and making edges visible in the picture, subtract it from the latest frame and keep doing this procedure to converge a constant number. It is obvious this procedure take great deal of times and energy and is not cost-efficient.
2- Another procedure which I adopt it in this project is calculating variance from pictures on which gradient applied. I keep doing this path to reach a constant value. It should be noticed that derivation of early frames could be equal. Hence, the variance rapidly reaches to constant number. Due to this fact, I take a more strike condition not to encounter with this kind

of problem. Therefore I take at least minimum frames to be sure about the result.
Another point is that if the algorithms don't reach to a reliable value and constant variance, automatically the procedure cease other part of the program and end the program.

The last part of my procedure is applying histogram on image that remains from the gradient demonstrated in Figure 3. An image histogram is a type of histogram which acts as a graphical representation of the tonal distribution in binary images. By adding the column elements of binary frames, the areas which are supposed as propagation of logos on horizontal axis is create. The histogram plot of Figure 3 is shown below

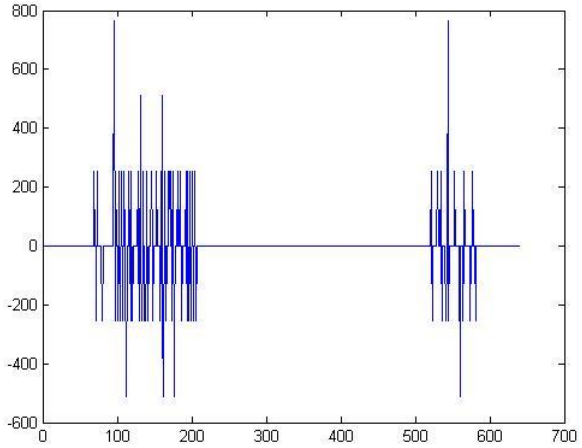

Figure 5: histogram plot of frame in which all of extra information is eliminated

To elucidate on, the histogram plots the number of pixels in the image with particular value. Propagate pixels either on vertical or horizontal axis led to emerge the margin of the logos.
When the variance of the frames remain constant during 10 frames (related to the experience), I stop processing the rest of the frames. Finally by detecting the coordinate of logos in videos, their specific portions will be deleted.

Figure 6: final picture

III. IMPLEMENTING ON GPU

Allocating host memory is by default pageable, that is, subject to page fault operations that move data in host *virtual memory* to different physical locations as directed by operating system. Virtual memory offers the illusion of much more main memory than is physically available.
The GPU cannot access data in pageable host memory because it do not have any authority to alter the memory which is controlled by CPU. When we are transferring data from host to device memory, CUDA by its very nature, allocates pinned memory in host and by then, start the copies procedures from pinned memory to device memory. Nowadays, the use of virtual memory is no necessary for many applications which could be fit in RAM. Hence, using pinned memory could easily accelerate the speed of data accessing and transferring.

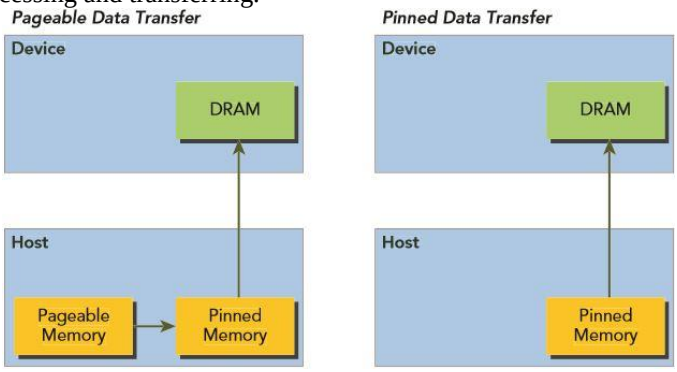

Figure 7: scheme of data transferring normally vs pinned memory

The first step in invoking the kernels which is related to the image processing and logo detection is allocating memory as pinned in host memory. Hence we are relieved from allocating memory in devices and tolerate the latency caused by data transferring. Another step toward program this project more conveniently is using zero copy memory. Generally, the host cannot directly access to memory of the devices and vice versa. To overcome with this problem, I declare a zero-copy memory in which either host or device could easily access with each other. According to my laptop which is integrated architecture in such a way that CPU and GPUs are fused in single die and their memories are near with each other, zero copy memory are more likely to benefit the performances and programmability, because no copies over PCIe buses are needed [5]. For discrete system in which devices connected to the host by PCIe, adopting this method could have deleterious effect on performance of program.

Due to programming the main function in C++ source file and using OpenCV, I should make a link with the *.cu* file. Therefore, I create extra .h file to make a link between these two files. Bunch of function which should be invoked sequentially, placed in main file and kernels functions are written in *.cu* file. To put it in a nutshell, CUDA functions do the processing in lieu of OpenCV built-in functions.
Another significant issue that should not go unnoticed is that all of the pixels in the images should be processed in parallel to cope with this problem. I generate the 2 dimension grid within 2 dimension block. Each thread in the blocks deal with

specific pixel. Beside of this fact, due to the independency between operations, all of the process could be implemented thoroughly parallel.

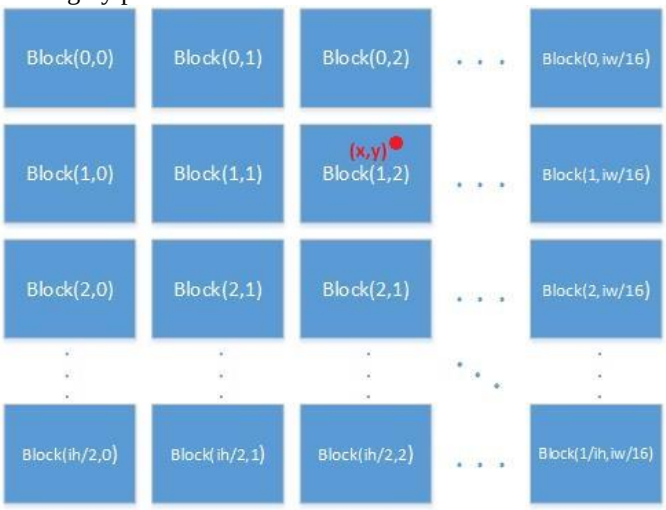

Figure 8: two dimensions of Grid which is suitable for image processing

IV. EXPERIMENTS & RESULTS

I ran all experiment in this report on a windows 10 with 2.3 GHz intel core I5 2410M CPU, 4 GB of memory Ram, and an NVidia GeForce GT 525M. The GPU program were compiled with the CUDA 7.5 toolkit. The IDE which I used is Visual studio 2013 updated 5. All results demonstrated on the rest of this reports, considers the transferring and allocating time either on GPU or CPU.

Performance summary: This project is 9.13 times faster than its purely CPU-based program in such a way that we reach to the same result with GPU in 3.12 seconds in comparison to its counterpart in which our program executed in 28.48 seconds. This shows my implementation is rational and using zero copy memory accelerates the execution time in an unparalleled rate.

Comparison with other journal: This project is based upon the real movie with whether high or low qualities in such a way that all samples which have been used to evaluate the accuracy and performance of my program, are downloaded 480p or lower qualities. To put it in other words, my program doesn't rely on any circumstances which could spoil the expected results. Beside of this fact the, latest journal related to this project was so slower. This project was compiled and executed with much weaker configuration system and reached to the better result. There isn't any constraint for the quality of videos for my projects. In this condition I've boosted the performance of the program 1.6x.

V. REFERENCES

[1] T.Albiol, "*detection of TV commercials",* IEEE international conference on (ICASSP), 2004

[2] Wojciech Jarosz, 2001, ACM SIGGRAPH, University of Illinois

[3] R. Fisher, S. Perkins, A. Walker and E. Wolfart. 2003, Sobel Edge Detector, The university of Edinburgh, Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

[4] M.Motamedi, R.Safabakhsh, 2014, "*A Fast, Parallelized Logo Detection Algorithm on Graphics Processing Units",* 22nd Iranian conference.

[5] J.Cheng, M.Grossman, T.McKercher, "*professional CUDA C Programming",* 2014, John Wiley & Sons, Inc.