ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 3, Issue 2, February 2018

A CUDA Based Implementation of Image Verification Tool Using Computer Vision Libraries

[1] Naveed Anjum Khan, [2] S. K. Kapde
[1][2]Deogiri Institute of Engineering & Management Studies, Aurangabad, Maharashtra State, India.
(Department Of Electronics and Telecommunication Engineering)

Abstract: **Image Verification has become major trend in automation industry to Validate images from high-resolution cameras at greater performance. Recent Technological developments in computing have made it possible for image processing algorithms to work at greater speed and at higher accuracy. However, in traditional serial manners, the operations of both methods are time-consuming. In this project, initially we are proving the possibility of CUDA based implementation of image comparison algorithms to automate image verification using NVIDIA CUDA Enabled Graphics Processing Unit, later we are going to compare the results with serial implementation and, finally a service executable based tool is developed using COM as well as named pipes based Inter - Process Communication interface to make these parallel algorithms accessible by third party tool like Lab view.**

Keywords: **- Automation, Validation, Parallel Processing, Graphics Processing Unit, CUDA, Inter Process Communication, performance comparison.**

I. INTRODUCTION

In imaging science, image processing can't abstain from being processing of images utilizing numerical operations by using any sort of standard managing for which the information is an image, a development of images, or a video, for event, a photograph or video outline; the yield of image processing may be either an image or a course of action of characteristics or parameters related to the image. Most image processing philosophy join seeing the photo as a two-dimensional signal and applying standard signal-processing to it. Images are in like manner managed as three-dimensional signals where the third-estimation being time or the z-axis. Image processing for the most part intimates digital image processing, however optical and basic image processing besides are conceivable. The getting of images (conveying the data image regardless) is recommended as imaging. Enduringly identified with image processing are PC delineations and PC vision. In PC representation, images are physically made using physical models of request, circumstances, and lighting, rather than being gotten (by technique for imaging contraptions, for occasion, cameras) from standard scenes, as in most breathed life into movies. PC vision, obviously, is a great part of the time saw as irregular state image processing out of which a machine/PC/programming to loosen up the physical substance of a photograph or a social event of pictures (e.g., recordings or 3D full-body appealing resounding channels). In cutting edge sciences and advances, images in addition grow considerably more wide degrees in light of the routinely making centrality of investigative acknowledgment (of as regularly as could be allowed unending scale complex sensible/exploratory information). Tests wire microarray information in acquired examination, or relentless multi-resource portfolio exchanging store [4].

II. LITERATURE REVIEW

Traditional methods for processing large images are extremely time intensive. Also, conventional image processing methods do not take advantage of available computing resources such as multi core central processing unit (CPU) and many core general purpose graphics processing unit (GP-GPU). Studies suggest that applying parallel programming techniques to various image filters should improve the overall performance without compromising the existing resources. Recent studies also suggest that parallel implementation of image processing on compute unified device architecture (CUDA)-accelerated CPU/GPU system has potential to process the image very fast. Author introduced a CUDA-accelerated image processing method suitable for multicore/manycore systems. Using a bitmap file, implement image processing and filtering through traditional sequential C and newly introduced parallel CUDA/C programs will run. A key step of the used algorithm is to load the pixel's bytes in a one dimensional array with length equal to matrix width * matrix height * bytes per pixel. This implement is to process the image concurrently in parallel. According to experimental results, the CUDA-accelerated parallel image processing algorithm provides benefit with a speedup factor up to 365 for an image with 8,192×8,192 pixels.[2] Each of the PC plans of the mid 1950s was a one of a kind configuration; there were no upward-perfect machines or PC designs with various, varying executions. Programs composed for one machine would not keep running on other kind, even different sorts of the same

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 3, Issue 2, February 2018

company. This was not a noteworthy disadvantage at the time in light of the fact that there was not a large body of software created to keep running on PCs, so beginning programming starting with no outside help was not seen as a vast hindrance. The design flexibility of the time was essential, for planners were exceptionally compelled by the expense of gadgets, yet simply starting to investigate how a PC could best be composed. A percentage of the essential components presented amid this period included list registers (on the Ferranti Mark 1), a return address saving guideline (UNIVAC I), prompt operands (IBM 704), and the discovery of invalid operations (IBM 650). Before the end of the 1950s business manufacturers had created industrial facility developed, truck-deliverable PCs. The most generally introduced PC was the IBM 650, which utilized drum memory onto which projects were stacked utilizing either paper tape or punched cards. Some top of the line machines additionally included center memory which gave higher speed. Hard plates were likewise beginning to end up prevalent [6] [12].

Before hyper-threading and multi-core CPUs came around, individuals tried to add extra processing criticalness to PCs by including extra CPUs. This requires a motherboard with more than one CPU alliance specific CPUs are embedded into various affiliations. The motherboard in a like way needs extra mechanical gathering to interface that CPU association with the RAM and unmistakable assets. There's an Enormous measure of overhead here, there's extra nonappearance of change if the CPUs need to exchange with each other, systems with various CPU will gobble up more power, and the motherboard needs more affiliations and hardware. This isn't particularly standard among home-client PCs today. Truly, even a handy gaming desktop with various representation cards will by and large basically have a single CPU. You'll discover multi-CPU system among supercomputers, servers, and top Notch systems that need as much figuring power as they can get [10]. Cache memory is the random access memory (RAM) that a microchip can get to this memory speedier than the general RAM. This memory is frequently planned with CPU or set on free chip that has an alternate transport to partner CPU. The key explanation behind cache memory is to store the system rule that is as regularly as could be allowed used by programming in the midst of operation. In view of this general execution extended. Consider a venture is continue running on a PC on different times the some rule or limit is secured in the cache memory. If the same task is sought after some time then cache memory recoups those limits which are secured in it and extension the pace of the execution [8].

III. PROPOSED SYSTEM

Image verification technology is widely used in automation industry; whereas current tools in this domain are utilizing only some part of hardware resource (Utilization of Single Core of CPU in contrast with the availability of Multiple Cores in CPU) available and leaving the other resource (Graphics Processing Units) unused. Considering this situation as a problem, we have formulated a solution to utilize the complete resource to reduce the burden on currently used resource, and to increase the performance by decreasing the time taken for each operation (verification). The current systems (DELL WORK STATIONS) in the company have a powerful graphics processing units (NVIDIA Quadro K2200) which are not utilized until some heavy graphics application uses it for rendering purpose, till then this resource is left unused. On the other hand OpenCV with CUDA support is providing a well-structured library to make image processing algorithms to run on GPU, hence, by executing these both conceives a new methodology, with this one can easily develop a tool that can utilize the power of parallel processing on GPU to reduce the time take for some algorithm like image filtering using convolution method where large amount of data need to be processed will be processed by GPU parallel instead of conventional serial manor. This has inspired us to propose this technology as a solution to the company. We have done one simple experiment to showcase the power of GPU over CPU as an illustration to company that this TOOL utilizes these concepts in its future implementation. The report of the experiment is as follows.

IV. MODULES

1. Template Matching:

•Read the images from the query_path and templ_path, *_global must be added while calling outines from COM.

•Upload the images from Primary Memory (RAM) to Video Card memory (VRAM)

•Perform Template Matching by calling the GPU API's provided by OpenCV in the namespace cv::cuda::

•Once the Matching is performed, the time structure takes the start time of the routine and end time of the routine to calculate the time taken by this routine

•Download matched image from VRAM to RAM and then to Secondary memory.

2. Intensity Measurement:

•Inputs for this routine is, query_path, the rectangle start coordinates and end coordinates x, y, xd, yd, *_global is used in the context of COM interface.

•Image is read from the query_path and intensity values are summed up in the ROI calculate from rectangular coordinates

then the summed up value is divided with pixel count to get average Intensity value of that region.
•If Java based GUI is used using pipe based IPC, Average Intensity Value is communicated to GUI.

3. Color Plane Extraction:
•Color Plane Extraction is a method of taking out the required Red, Green or Blue plane out of the BGR color Image
•This routine has the input parameters that will take query_path and channels required and gives out a cv::Mat object which is a color plane needed.

4. Image Transformations:
•Crop, resize, rotate and Mask are the basic image geometrical transformation that can be applied on image.
•OpenCV provides enough functions to deal with geometrical transformations
•The Figure 5.4 is self-explanatory; the rotate function is supported with warp affine function to create a background during rotation by angle.

5. Frame Grabbing:
•Frame grabbing is a method developed to extract required frame, range of frames and all the frames from the video.
•As all routine this routine has COM as well as pipe based IPC interface Thought command parameter in this method the single frame, range of frames and grabbing all frames is controlled.
•All the extracted frames are stored as per the location provided by the user to this method using Java GUI or COM client.

6. Shape Detection:
•The Open CV inbuilt routine namely Hough Circle and Hough Lines are used to detect the shapes in the image
•Initially the image is converted into gray scale Canny edge detection is performed on the image
•Later Hough Circle and line detection is performed on the image to determine the location of circles and lines
•Finally the response image is plotted with circles and lines

7. Geometry:
•To calculate the distance general Euclidian distance method is used and to calculate the angle the general Math technique is used to solve the two coordinate geometrical calculations
•On the other side the image need to be loaded from secondary storage to primary to get the dimensions of the image
•And to verify the input coordinates doesn't exceed the dimensions of the image.

V. CONCLUSION AND FUTURE WORK

The results for four graphics cards for different size of images are shown previously. By seeing the results we can conclude that for NVIDIA GF119–300–A1 which have 46 CUDA cores, the performance of CPU is good as compare to GPU for low resolution image i.e. nHD. But as the image size increases to higher resolution i.e. for qHD and HD the performance of GPU is increases in good manner. For NVIDIA GF108–400–A1 the results for low resolution i.e. nHD are comparatively same. But for higher resolution like qHD and HD GPU performance is better than the CPU. For NVIDIA GM107–400–A2 the performance of GPU for low resolution and as well as high resolution is far better than previous two graphics cards. For NVIDIA GM107–300–A2 the results are more impressive than other graphics card. GPU timing got boost by using this graphics card because it has 640 CUDA cores. So this work concludes that higher GPU shows performance edge over CPU both in high-end and low-end GPUs, on the other hand, a low-resolution image has no performance improvement on Low-end GPUs compared to CPU, but on a high-end graphics card like K2200 low-resolution shows greater performance edge over CPU.

REFERENCES

[1] Yuehu Liu, Bin Chen, Hao Yu, Yong Zhao, Zhao, Zhou Huang and Yu Fang, "Applying GPU and POSIX Thread Technologies in Massive Remote Sensing Image Data Processing", National Project Of Scientific and Technical Supporting Programs Funded by ministry of Science & Technology of China, 2011, pp.3-7.

[2] Abu Asaduzzaman, Angel Martinez, and Aras Sepehri "A Time-Efficient Image Processing Algorithm for Multicore/Manycore Parallel Computing", IEEE Southeast Con 2015, April 9 - 12, 2015 - Fort Lauderdale, Florida.

[3] Shalini Gupta, Senior Mobile Computer Vision Engineer, Nvidia. OpenCV - Accelerated Computer Vision Using GPUs (n.d.): 5-15. Web.

[4] "Image processing", Online Available: https://en.wikipedia.org/wiki/Image_processing, 6th April, 2016.
"Digital Image Processing", Online Available: https://en.wikipedia.org/wiki/Digital_image_processing, 7th April, 2016.

[5] "Template Matching", Online available: http: //docs.opencv.org/2.4/doc/tutorials/imgproc/ histograms/template_matching/template_matching.html, 8th April, 2016.

[6] "Central Processing Unit", Online Available: https://en.wikipedia.org/wiki/Central_processing_unit, 10th April, 2016.

[7] "Graphics Processing Unit", Online Available: https://en.wikipedia.org/wiki/ Graphics_processing_unit, 11th April, 2016.

[8] "Cache Memory", Online Available: http://searchstorage.techtarget.com/definition/cache-memory, 11th April, 2016.

[9] "Linux SMP", Online Available:http://www.ibm.com/developerworks/library/l-linux-smp/, 11th April, 2016.

[10] "Multi-Threading with Microsoft", Online Available: https://msdn.microsoft.com/en-us/library/y6h8hye8.aspx,12th April, 2016.

[11] "NVIDIA GeForce 610", Online Available: http://www.geforce.com/hardware/ desktop-gpus/geforce-gt-610. 12th April, 2016.

[12] "NVIDIA GeForce 630M", Online Available: http://www.geforce.com/hardware/notebook-gpus/geforce-gt-630m. 12th April, 2016.

[13] "NVIDIA GeForce GTX 750TI", Online Available: http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-750-ti, 12th April, 2016.

[14] "NVIDIA QUADRO K2200", Online Available: https://www.pny.com/nvidia-quadro-k2200?sku=VCQK2200-PB&type=m,12th April, 2016.

[15] "Open CV Documentation", Online Available: http://docs.opencv.org /3.1.0/#gsc.tab=0, 14th April, 2016.

[16] "CUDA Documentation", Online Available: http://www.nvidia.com/object/cuda_home_new.html, 14th April, 2016.

[17] "Open CV with CUDA Documentation", Online Available: http://opencv.org/platforms/cuda.html, 17th April, 2016.

[18] "Modern GPU", Online Available: http://nvlabs.github.io/moderngpu/, 18th April, 2016.

[19] "NVIDIA GeForce 630M", Online Available: https://www.tele-task.de/archive/podcast/11324/, 20th April, 2016