TRACKING OF RIGID BRIGHT COLOR TOY IN HETEROGENEOUS COMPUTING SYSTEMS USING OPENCL

Devdutt Baresary
Assistant Professsor
Department of Computer Science and Technology
Gulzar Group of Institutes, Ludhiana, India.

Abstract: Image processing is widely used in many computer games, modern toys, defense, etc. We know that our image like, Gray scale images or Colored images both have complex data, the image pixels. When we have to make the histogram of an image the sequential algorithm waste more clock time. Use of parallel programming have a capability to reduce time taken of algorithm as compared to CPU at the same time it increase throughput. Our work is to use a the object and reconstruction it in 3D plane. The histogram thresholds value are helpful in segmenting the blocks and merge them into one resulting components but this process takes lot of time because the image that we capture have high processing time. Here we are using the concept of parallelism for decreasing the time taken. We are presenting the image histogram algorithm by using parallel processing environment using OpenCL.

Index Terms: OpenCL, OpenCV, gnuplot, sequential architecture, parallel architecture, GPU.

I. INTRODUCTION

The tracking of rigid bright color toy project investigates to design a smart specification for object tracking applications using OpenCL language. Simple implementation by using sequential algorithms take more CPU time but by using parallel processing technique which decrease the processing time for an image processing application. By use of coarse grained concept the problem is divided into discrete chunks which executed at the same time [1]. By using this parallel computation we can execute multiple instructions simultaneously. Using high level languages the algorithms are implemented sequentially we are also using languages like C, Matlab, OpenCV. When we are executing the sequential algorithms on a CPU it runs slower. In the proposed system the sequential algorithms which have task parallelism or data parallelism those algorithms are converted to OpenCL.

By the help of OpenCL we minimizes overhead on the CPU and makes histogram and other image processing algorithms run faster and achieving 25x, 39x and 50x higher throughput [2].

Concurrency is the way to sharing of multiple resources in a hardware. The dependencies in the system in different components is highly inefficient and lower the overall performance of the system. The performance of the system is furnished by the use of parallel processing concepts by which we divide our problem and executed simultaneously on different compute units.

Maximizing the effectiveness of the system concurrency provides the most efficient way. When the multiple threads running in parallel, this means that the active thread running simultaneously on different hardware resources and processing elements. The execution of the simultaneous threads is provided by the platforms, for achieving parallel computing.

In latest computing machines we have SIMD or MIMD architectures for parallel programming which have capabilities to utilize either data level or task level parallelism [6].

- **Task level parallelism,** to accomplish different number of tasks, within a single problem at the same time. The proficiency of this model will depend on the independent operations of the task [6].
- **Data level parallelism,** to manage the isolated portions of the same task at the same time. The proficiency of this model will depend on the independent operations of the task [6].

II. DEFINING THE ALGORITHMS

The parallel processing algorithms are proposed for bright color rigid toy. The representation of algorithms are presented in Figure 1.The first step is bilateral filtering[7], we have input as an image rich by color inhomogeneity which eliminates by applying bilateral filtering. Bilateral filtering has capabilities to remove the blur at the same time it preserve the edges. The images have complex data in the form of pixel values when we applying the bilateral filter on an image it takes time to execute on CPU. But if we are applying the parallel processing concepts then positively we reduce time taken by the sequential bilateral filtering algorithm.

Second step, is color histogram of an image. We have to make a histogram of that image that we obtained from first step. The threshold value of histogram which we call as histogram peak value is being used for object segmentation in third step.

Third step, segmentation of blobs [8] by using histogram threshold value from second step. The object consists of color blobs are integrated in one connected components. Every block in the image is represented by the connected components.

Fourth step, each connected components drawn-out to its neighbor that represents the whole object after merging them into one connected component. We are applying this dilatation procedure on the image various times. The dilatation procedure is work for covering the edges between blobs for that we choose the largest connected components in an image.

Fifth step, the intersection of the points or we can say that features are not specific by the feature point algorithm. Which create noise, to make it correct we have to use corner detection algorithm to the each neighborhood locations. Near the guess point we get the sub-pixel point position.

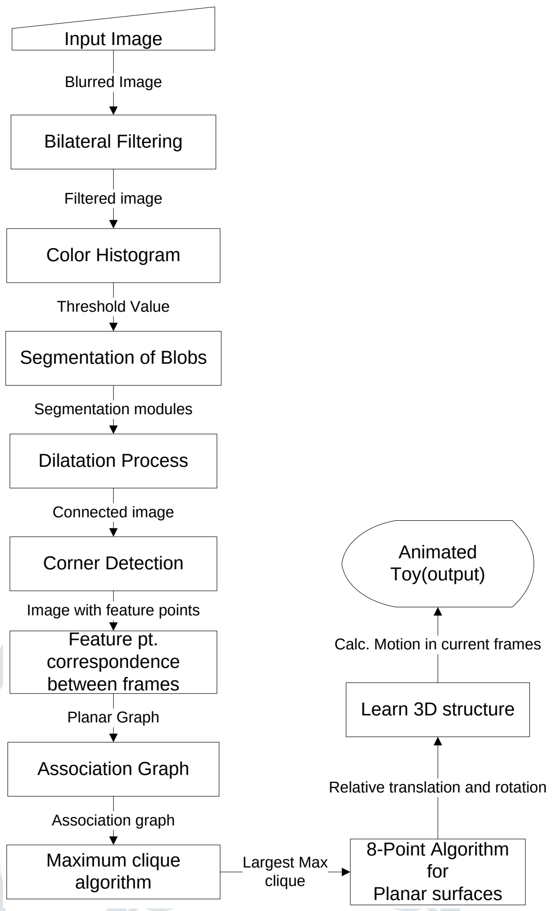

Fig. 1: Flow chart of algorithms

Sixth step, between the two frames the feature point correspondence are achieved in two steps. First one is to create correspondence between blobs and consequently neighboring each blob match the feature points. We construct the sub-graph isomorphism for the correspondence of the frames. For that we have to build-up planar graph for each frame.

Seventh step is creating association graph for verification of the two isomorphic graphs we have.

Eighth step, using maximum clique algorithm in the association graph we find the largest maximum clique in the largest sub-graph which have all the vertices already connected to each other.

Ninth step, as we have a rigid object with established correspondence between the frames. We have to use eight point algorithms for reconstruction of a toy in the occurrence of sequence of monocular views. The algorithm evaluate fundamental matrix from a set of eight or more corresponding image points. Actually, the image coordinates of the image positions affected by distortion. The object is planar, so the algorithm has unsatisfactory results because same plane consist of all feature points. The resulting output is relative translation and rotation of the rigid object between two frames.

Tenth step, because of the relative translation and rotation between the two related frames is relatively small. Sometime it produces unwanted errors in 3D reconstruction because of the change in position becomes smaller caused by the noise in the image. For diminishing effect of noise in reconstruction, to use for compute action in the current frame we have to pick up 3D structure of the object from the prior frames.

Eleventh step, for identifying the each object we need the matching function and 3D geometry evaluation. For acquiring the largest maximum clique that is equivalent to the association graph. For recognizing the object subsequently we have to collect the blob graph with the 3D geometry captured from the series of frames. We extract the largest maximum clique which is equivalent to the association graph in the pair of frames of the related object.

We have to follow two different strategy related to the algorithm for identification, search the largest maximum clique when we compare the blob graph of the test image to the each saved blob graph. Thereafter we calculate the maximum size clique in the saved blob graph.

III. IMPLEMENTING COLOR HISTOGRAM USING OPENCL:

Mostly the histogram is the illustration of the distribution of the colors in an image. Histogram is the comparison between the pixel counts and pixel intensity values. Color histogram is made for any kind of color space, like intensity, RGB, HSV, CYN, etc. Color histogram plots the number of pixels in the image (X axis) with the particular brightness value (Y axis).Image histogram can be evaluated for the peak and/or valleys later on which can decide the threshold value.

The histogram is the well-defined concept which is helpful in calculating the threshold value of the histogram which we called as the histogram peak value. The series of histogram threshold value is applied to the image for the image segmentation with the upcoming detection of the large connected components.

The histogram algorithm we are taken here is for the RGB values. Which having R as red, G as green and B as blue. The algorithms calculate the histogram as red followed by green followed by blue. The source data for this procedure is an 8-bit-per-pixel-image with the target of calculating the histogram bins of 32 bits into each 256 counts.

The image is split into number of workgroups, as per the coarse grained concept because if we go for fine grained the hardware is not support that. The workgroup is the part of an image by which we make the sub-histograms stored in the on chip local memory of the GPU. These partial histograms are subsequently compact into the single main histogram which we call global histogram

Overview is as follows, when we originate to propose a OpenCL kernel to the corresponding hardware we have to take care of some restrictions on the number of work-items and the size of the workgroup. Local memory is existing in the workgroup and we communicate with the data within the workgroup. If we want to share the results of the each workgroup to another workgroup we are not allowed for this. The workgroup comprises of work items which share amongst the local memory inside that workgroup. The partial histogram is accumulated in the local memory we are doing all this because of degrade memory burden.

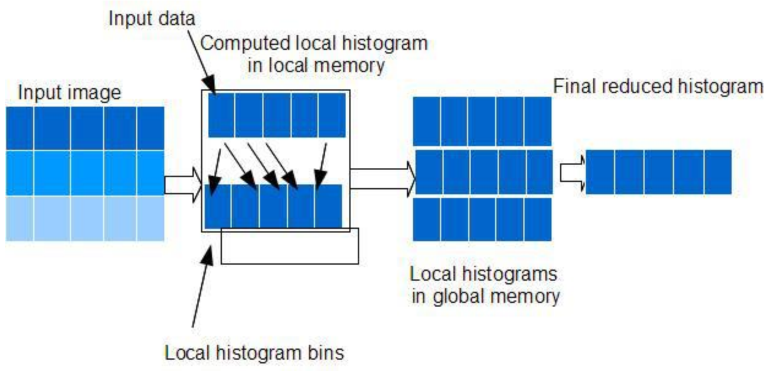
Input data
Computed local histogram in local memory
Input image
Final reduced histogram
Local histograms in global memory
Local histogram bins

Fig 2**:** Basic idea of distribution of image as output data decomposition.

We have to generate more workgroups for assigning more local memory data into the global memory but we have to make sure of the number of workgroups for effective usage of local memory and reduce burden on the global memory the number should be close to the number of compute units we have in our hardware.

To finding the workgroup size (128) for this problem is difficult to find we selected at minimum the even multiple of the total size of the compute units we have.

To traverse the total input image we use here, global Ids, local Ids, group Ids, group size, etc.

Our OpenCL program consists of two parts: First is kernel part, the instances of kernel are copied to the different compute units and the kernel is executed on the each compute unit individually. Another is host code which we have to add to work for the different models of the OpenCL which is executed on the host or CPU.

The host program defines the context for the kernels and manages their execution. Every OpenCL device has a command queue, where the host programs are queued for kernel execution and memory transmission.

The fundamental of the OpenCL execution model is expressed by execution of kernels. When kernel is executed an index space is defined which is an instance of the kernel. As in our problem we have two kernels: First one is for calculating the partial histogram of the part of an image. Next one is for merging the partial histograms into global histogram results. This index space we describe as ND Range kernel. The instance of the kernel is called work item. Each work item is identified by a global Id which is he index space in ND Range kernel. Each work item executes the same code but the execution route and the data used will be altered.

Each workgroup is assigned a unique workgroup Id and each work item in the workgroup assigned the local Id. The work item in a single workgroup executes concurrently on a processing elements of a single compute units. Work item in a workgroup can coordinate each other and distribute data across local memory in the compute unit. All the work items has read and write access to any position in the global memory. The global and constant memory can also be accessed from the host processor before and after kernel execution.

IV EXPERIMENTS AND RESULTS:

For each image histogram algorithm, both the GPU kernel code and CPU serialized code are designed. The pixel values we are taken into account are in unsigned normalized 8-bit integer value (unorm8) to ensure comparability of the results. The kernel code is copied to the various compute units and the resultant of the code is stored in the CPU memory.

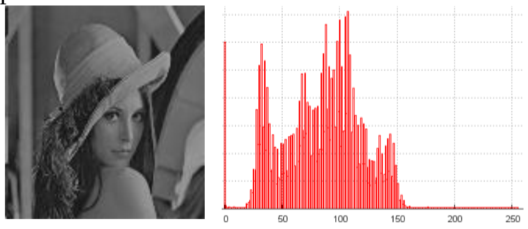

Fig. 3**:**Image histogram(128 * 128) serial code for gray scale image

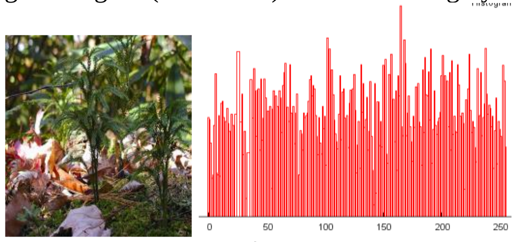

Fig 4**:** Imag**e** histogram (640 * 480) for RED bins made by using OpenCL.

The GPU is significantly improves the computing speed as the image size is increase. We are using here PNG color image, the algorithm is capable of computing the histogram for other image formats too. The size of the image ranges from (128*128) to (1920*1080). The speed up is increased by the increment of the image size as the GPU work well with more complex data. On GPU/OpenCL algorithm takes 143.34 ms approximately.

V. CONCLUSION:

In this paper one image histogram algorithm is presented and implement by using OpenCL environment and compared with the sequential implementation on CPU. The result of the algorithm is same as the histogram created by the faststone image viewer for the same images. So,the algorithm take less time on GPU more than 40x speed up rather than CPU. Future work is to gain profound knowledge about the parallelization techniques to make best use of GPU device and to work with other algorithms associated with this domain.

References:

[1] "*Heterogeneous computing with OpenCL*" by Benedict gaster,british libraries,printed USA

[2] http://www.khronos.org/OpenCL ,"Khronos Group".

[3] Nan Zhang, Yun-shan Chen, Jian-li Wang. "*Image Parallel Processing Based on GPU*". International Conference on Advanced Computer Control, March 2010.

[4] Pardalos P.M., Xue, J, "*The maximum clique problem*",Journal of Global Optimization, 4, 1994, 301—328

[5] http://developer.amd.com/pages/default.aspx *"University Kit 1.0".*

[6] Tomasi C., Manduchi R., "*Bilateral filtering for gray and color images*", ICCV, 1998, 839-846.

[7] Skarbek W., Koschan A., "*Color image segmentation a survey*". Technical report, Technical University, Berlin, 1994.

[8] *AMD Accelerated Parallel Processing OpenCL Programming Guide1.pdf*

[9] Shi J., Tomasi C., "*Good features to track*". CVPR, 1994, 593-600.