Parallelization and Optimization of Pedestrian Detection Software on NVIDIA GPGPU using CUDA-C

A.D. Londhe
K.V. Bhosale
Pune Institute of Computer Technology,
S. No.27, Dhankawdi, Pune

Sayli Zope
Roshani Rode
student

Rasika Waichal
Rajat Toshniwal
student

ABSTRACT

The future of the computation is the Graphical Processing Unit, i.e. the GPU. The graphics cards have been shown in the pasture of image processing and accelerated interpretation of 3D scenes, and computational capability that these GPUs acquire, they are rising into immense parallel computing units. It is quite simple to program any graphics processor to execute universal parallel tasks. But after understanding the various architectural features of the graphics processor, it can be used to execute other demanding tasks as well. In this paper, the use CUDA (Compute Unified Device Architecture) can fully utilize most tremendous power of these GPUs. CUDA is most popular parallel computing architecture of NVIDIA. It enables dramatic increases in computing performance, by harnessing the power of the GPU. In this, the sequential software of the famous image processing problem, the pedestrian detection[1] and parallelize it to run on GPUs and increase the performance of that software. We have presented a new pedestrian detector that improves in speed and quality of the pedestrian detection. By efficiently handling different scales and transferring computation from the test time to training time, detection speed is improved.

General Terms

GPU Computing, Histograms of Oriented Gradients(HOG) algorithm

Keywords

Pedestrians Detection, Compute Unified Device Architecture(CUDA), Object Detection, Graphics processing units (GPUs), Parallel Architectures,SVM Classifier.

1. INTRODUCTION

Graphics cards are mainly used to accelerate the gaming and the 3D graphics applications. The GPU of a graphics card is built for to compute intensive and highly parallelized computations. With the prevalence of high level APIs (CUDA), the power of GPU is being leveraged to hasten more general purpose and soaring performance applications. It has been used in the accelerating database operations, solving differential equations, and geometric computations. Image processing is well-known and most established research field. It is the form of signal processing in which the image is an input, and the output can be an image or anything else that will be some meaningful processing for work. [2][1]Altering an image to be brightens, or darker is an paradigm of a regular image processing tool that is available with basic image editors. Often, processing happens on the complete image, and the similar steps are useful to every pixel of the image. There is a lot of reputation of the same work. Newer technologies allow better quality images to be taken. This equates to bigger

files and longer processing time. With the advancement of the CUDA, programming to the GPU is simplified[4]. The technology is ready to be used as a problem solving tool in the field of image processing.

Among the Advanced Driver Assistance Systems i.e ADAS system, pedestrian detection is the common issue due to the vulnerability of the pedestrians in the event of accidents. Pedestrian is nothing a person walking rather than traveling in a vehicle. Pedestrian detection is a very essential and significant task in any intellectual video examination system, as it provides the essential information or semantic understanding of the capture footages. It has an noticeable expansion to automotive applications for improving safety systems for use. Pedestrian detection is a important part of object detection. Because of its direct applications in car safety, surveillance, and robotics, it has fascinated much concentration in the last years..

2. OVERVIEW

2.1 Related Work

There is an extensive literature on object detection, but here we mention a relevant paper on human detection. Of the several approaches that has been proposed for detecting pedestrians, one common method uses a pre-trained classifier within a sliding window to scan the complete image looking for community at all locations and scales[2]. High quality implementation of this is a major challenge today and the computer industries struggle with how to efficiently engineer this. There is need to work on humanizing uncovering speed (with no trading-off quality) exploits one or more of the following ideas:

2.1.1.1 Object features

Having cheap to compute the features that capture best the input image information, is the crucial for fast and good detections[1].

2.1.1.2 Object classification

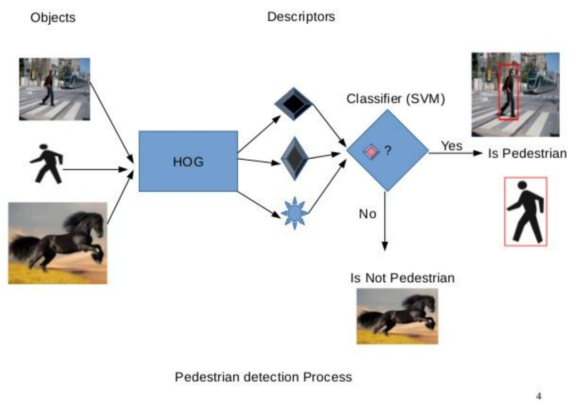
For a given set of features, the choice of classifier has impact on the resulting speed and the quality, often requiring the great trade-off between these two. Non-linear classifiers will provide the best quality,[1] but suffer from low speed. As a result, linear classifiers such as the Adaboost, then linear SVMs, or Random/Hough Forests are more commonly used.

2.1.1.3 Better prior knowledge

In general, image processing is greatly benefits from the prior knowledge. For pedestrian detection the presence of a single

dominant ground plane has regularly been used as prior information to recover both speed and quality.[1][2][6]
As the above techniques are implemented sequentially, these are not best suited for real-time applications. To make it suitable for real-time usage, these techniques need to be parallelized. Implementing such systems raise complex parallelism and scheduling issues. Also when using GPUs for obtaining parallelism, optimization remains an open problem. Fig.1 shows the process of pedestrian detection.

Fig 1:



Pedestrian Detection Process

2.2 Pedestrian Detection

The evolution from the baseline CPU detector running at 0.08 Hz up to GPU detections at 135 Hz. Speed-wise results will be more faster than that reported earlier. Also our result will be the more than 10 times faster than the cudaHOG results reported from the GPU implementation. Fig.2 shows the pedestrian detection in image.

2.2.1.1 *Speed measurement*

We measure the speed taken by the CPU+GPU starting when the image frames are available both to the CPU and the GPU. [4]The measured time,does include all the CPU and GPU computations and the time[5] to download the GPU results and to run the non-maximum suppression on the CPU. As previously indicated our desktop computer is equipped with an Intel Core i7 and an NVIDIA GTX Titan X.

In this, the GPU based implementation contained in the OpenCV library ([2]) to speed up the feature computation process, and an own GPU implementation of the sliding-window SVM classification, where we took care that the entire feature computation and classification tool-chain is conducted in GPU memory.

1) Pedestrian detection: While common approaches use linear SVMs for detection, our approach makes use of more powerful (but slower) non-linear SVMs as well, arranging in the form of a detection cascade. This allows us to circumvent the speed disadvantage of non-linear SVMs as they are only applied to the (few) detections given by the linear SVM stage.

2) Pose classification: Hypotheses who have been approved both by linear and non-linear SVMs are subjected to pose classification using a set of K pose-specific non-linear SVMs. After training, pose classification is conducted using pedestrian images provided by the real-time system.

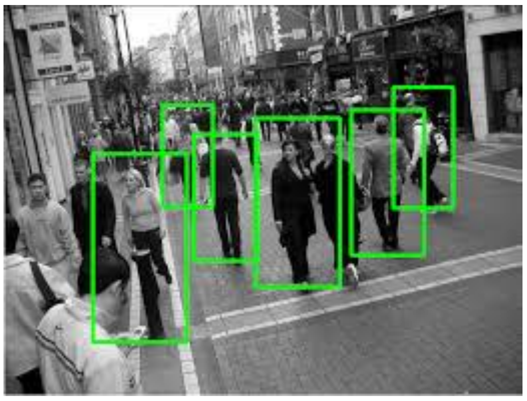


Fig 2: Pedestrian Detection

2.3 GPU Implementation

2.3.1.1 *Basic Understanding*

In GPU programming, mostly the GPU code and the input data typically need to be copied from the host to the device memory before we launch it on the GPU kernel, [5]on the GPU. The output data will also be copied back from the device to the host memory, so that the CPU can read them efficiently. Hence the GPU computation[4] comes at the huge expense of the offloading overhead on code. Another shortcoming of the GPU is it has relatively low operating frequency.

We use CUDA mostly for GPU programming. A unit of code that is independently launch on the GPU is called as a kernel. The kernel is made of multiple threads that execute the code in the parallel. [4][5]A unit of threads that are scheduled by the hardware is called a block, while a compilation of blocks for the analogous kernel is called as a grid. The maximum number of threads that can be contained by an individual block is defined by the GPU architecture.

2.3.1.2 *Program Analysis*

The usage of the GPU is depends highly on the program structure. If the program does not contain data-parallel and compute-intensive blocks, the GPU is not that much effective at all. Therefore it is important to analyze the program prior to the coding and the[1] implementation. The following is the summary of the program sequence for HOG-based object detection using the available deformable models used. The detailed procedure and algorithm description are:

1) Load an input as image.
2) Load the pre-defined object models of that image.
3) Calculate HOG features for all resized variants of the input image, often referred to as a HOG pyramid.
4) Calculate the similarity scores for every set of the root or the part of the filters and the resized HOG images.
5) Detect the object region based on a summation of the similarity scores.
6) Recognize the object which is based on the detection of result.
7) Output the recognition result.

"Support Vector Machine" (SVM) is the supervised machine learning algorithm. SVM algorithm which can be used for both the classification or regression challenges. In this algorithm, we plot every data item will point in n-dimensional space where n is number of the features [9]6][you have by way of the value of each feature being the assessment of the exacting organize. Then, we perform the classification by judgment the hyper-plane that discriminate the two classes in very well (look at the snapshot).Support Vectors are the coordinates of an individual observation. Support Vector Machine is the frontier which is the best segregate the two modules (hyper-plane/ line).

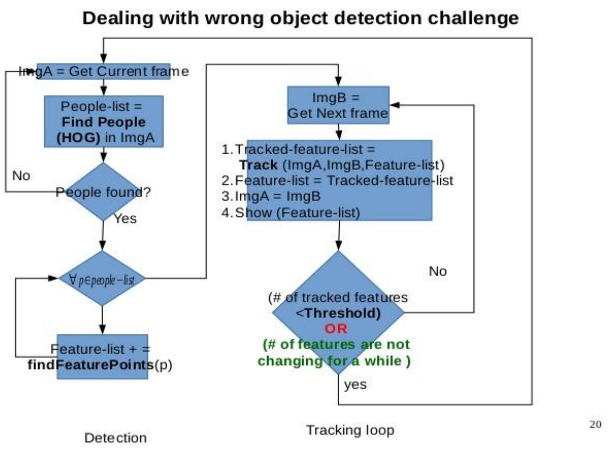

Dealing with wrong object detection challenge
ImgA = Get Current frame
People-list = Find People (HOG) in ImgA
People found?
No
Yes
∀ p∈people−list
Feature-list + = findFeaturePoints(p)
ImgB = Get Next frame
1. Tracked-feature-list = Track (ImgA,ImgB,Feature-list)
2. Feature-list = Tracked-feature-list
3. ImgA = ImgB
4. Show (Feature-list)
(# of tracked features <Threshold)
OR
(# of features are not changing for a while)
No
yes
Detection
Tracking loop
20

Fig 3: Wrong object detection algorithm

3. CONCLUSION

We have presented GPU implementations of pedestrian detection using parallel HOG and SVM algorithms. Given that this performance improvement is obtained from the entire program execution rather than a particular algorithm within the program, this is the significant contribution for the real-world applications.

In future work, we plan to achieve high level optimization. Although our current HOG and simple linear SVM detector is reasonably efficient processing in less than a second, there is still room for optimization and to further speed up detections. It would be useful to develop a coarse-to-fine or rejection chain style detector based on HOG descriptors. We are also working on HOG-based detectors that incorporate motion information using block matching or optical flow fields.

4. ACKNOWLEDGMENTS

We are grateful to our internal guide Prof.A.D.Londhe and Prof.K.V.Bhosale for giving us all the help for project and guidance we needed. We are really grateful to them for their kind support. Their valuable suggestions and guidance was very helpful.

We are also grateful to Prof. G.P.Potdar, Head of Computer Engineering Department, PICT for his indispensable support, suggestions. In the end our special thanks to Lab Teachers for providing various devices and resources such as laboratory with all needed software platforms, continuous Internet connection, for Our Project.

5. REFERENCES

[1] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In Proc. of the IEEE Conference on Compute Vision and Pattern Recognition, pages 886–893, 2005.

[2] S. H. U. Chang, D. Xiaoqing, and F. Chi, Histogram of the Oriented Gradient for Face Recognition , Tsinghua Science and Technology, vol. 16, 2011, no. 2, pp. 216-224.

[3] R. Benensonetal. Pedestrian detection at 100 frames per second. In CVPR, 2012. P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminately trained part-based models. Pattern Analysis and Ma-chine Learning (PAMI), Sept. 2010.

[4] NVIDIA. NVIDIA’s next generation CUDA computer architecture:Kepler GK110. http://www.nvidia.com/, 2012.

[5] A. Danalis, G . Marin, C. McCurdy, J.K. Spafford, V. Tipparaju, and J.S. Meredith, P. C. Roth, S. Vetter. T he Scalable Heterogeneous Computing (S H O C) benchmark suitings of the 3rd In Proceed Workshop o n General-Purpose Computation on Graphics Processing Units, GPGPU ' 1 0, pages 63-74, New York, NY, USA, Mar. 2 0 1 0 . ACM

[6] K. Mizunoetal. Architectural Study of HOG Feature Extraction Processor for Real-Time Object Detection. In SiPS, 2012.

[7] P. Viola, M. J. Jones, and D. Snow. Detecting pedestriansusing patterns of motion and appearance. The 9th ICCV, Nice, France, volume 1, pages 734–741, 2003.

[8] Z. Song, M. Wang, X. Hua, and S. Yan. Predicting occupation via human clothing and contexts. In ICCV, 2011.

[9] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation ofthe state of the art. PAMI, 34(4):743–761, 2012.