GPU-Based Complex-Background Segmentation Using Neural Networks

Dubravko Ćulibrk[†] and Vladimir Crnojević[‡]
Faculty of Technical Sciences
University of Novi Sad
Trg Dositeja Obradovića, 21000 Novi Sad, Serbia
dculibrk@uns.ac.rs[†], crnojevic@uns.ac.rs[‡]

Abstract

Moving object detection in visual surveillance videos is usually accomplished through foreground-background segmentation. When segmentation of interesting moving objects from the background is to be done in the presence of moving objects in the background itself, the process calls for the use of complex probabilistic background models. Maintaining such models is computationally intensive and limits the real-time applications of such methodologies to low resolution sequences, far below the acquisition ability of state-of-the-art cameras. While most probabilistic foreground segmentation approaches can benefit from parallel processing, since they allow for parallelization of operations done for each pixel, the Background Modelling Neural Networks (BNNs) allow the process to be parallelized even further, at the level of feature patterns stored in the model for each pixel. The paper presents a parallel implementation of BNN based moving object segmentation, running on NVIDIA^{TM} Graphics Processing Unit (GPU). Several modifications to the original algorithm are proposed to make for an efficient implementation. Experiments show that the approach is able to achieve real time processing of Standard Definition (SD) video in real time, never before reported for a probabilistic approach.

1 Introduction

Object segmentation is a fundamental computer vision problem and plays a key role in various applications. Moving object segmentation in particular is extensively used in automated surveillance [5][10][14], which is the focus of the work presented here.

Over the past 25 years significant research effort has been spent on developing ways of segmenting the moving foreground objects from the background. When video is captured from a stationary camera, which is a common assumption in surveillance, the background is expected to be stationary to a degree and an adaptive model can be built to serve as basis for segmentation. A special class of hard segmentation problems is concerned with applications dealing with natural-scene sequences with complex background motion (e.g. trees moving in the wind, rippling water, etc.) and changes in illumination, where the stationary background assumption is seriously compromised. When this is the case, state-of-the-art published solutions rely on multi-modal probability estimates of pixel values occurring in the background, as the background model, and statistical tests to determine the likelihood of a pixel belonging to the estimated probability distribution (i.e. background) to achieve foreground segmentation [5][6][10][14]. Unfortunately, the process of building and maintaining probabilistic models and performing segmentation in this manner is computationally expensive and allows for real-time segmentation to be performed only for low resolution videos, typically QCIF or CIF [10][5]. What is more, real-time performance cannot be achieved for more complex kernel-based methods [6][13].

GPUs are especially well-suited to address problems that can be expressed as data-parallel computations (the same program is executed on many data elements in parallel) with high arithmetic intensity (the ratio of arithmetic operations to memory operations). Because the same program is executed for each data element, there is a lower requirement for sophisticated flow control; and because it is executed on many data elements and has high arithmetic intensity, the memory access latency can be hidden with calculations instead of big data caches [1]. Many computer vision operations map efficiently onto the modern GPU, whose programmability allows a wide variety of computer vision algorithms to be implemented [9].

Background Modelling Neural Networks (BNN) represent a recently proposed kernel-based foreground-background segmentation methodology designed specifically with a parallel implementation in mind [5], as the only way to achieve real-time segmentation of high-resolution complex natural-scene sequences. The approach exploits

inherent traits of neural networks to enable parallelization of the process at a sub-pixel level. Here an NVIDIA^{TM} Graphics Processing Unit (GPU) based implementation of the BNN approach is described which allows for real-time processing of Standard Definition (SD) sequences. Several modifications of the original algorithm are proposed to make for an efficient implementation. The implementation described relies on NVIDIA Compute Unified Device Architecture (CUDA) [1] and OpenCV [3] technology. The result has been tested on a large set of sequences standard for the domain in question. the implementation achieves segmentation of 10 frames per second (fps) for SD sequences and over 60 fps for (160×128 pixel) sequences, including the time it takes to load the video frames and display them.

2 RELATED WORK

Probabilistic techniques represent the state-of-the-art in foreground-background segmentation. They estimate the statistics of the pixels corresponding to background and use them to distinguish between the background and the foreground[5][14][13][10]. A Gaussian-based statistical model whose parameters are recursively updated in order to follow gradual background changes within the video sequence has been proposed in[2]. More recently, Gaussian-based modelling was significantly improved by employing a Mixture of Gaussians (MoG) as a model for the probability density functions (PDFs) related to the distribution of pixel values. Multiple Gaussian distributions, usually 3-5, are used to approximate the PDFs [14]. MoG approach of Stauffer and Grimson [14] is probably the most widely used segmentation approach to date. Several improvements to their original methodology have been made over the years [15].

Gaussian-based models are parametric in the sense that they incorporate underlying assumptions about the shape of probability density functions (PDFs) they are trying to estimate. This can lead to a rough approximation of the (PDFs) and impact their performance [10]. Nonparametric models have been receiving significant attention in recent years. A nonparametric kernel density estimation framework for foreground segmentation and object tracking for visual surveillance has been proposed in [6]. The authors present good qualitative results of the proposed system, but do not evaluate segmentation quantitatively nor do they compare their system with other methods. The framework is computationally intensive as the number of kernels corresponds to the number of observed pixel values. In 2003, Li *et al.* proposed a nonparametric method for foreground object detection [10], which represents a hybrid between the filter-based and probabilistic methodologies. The primary model of the background used by Li *et al.* is a background image obtained through low pass filtering. The authors use a secondary probabilistic model for the pixel values detected as foreground through frame-differencing between the current frame and the reference background image. Their model is a histogram of pixel-values occurring at a pixel location.

The approach based on background modelling neural networks was proposed in [5]. The networks represent a biologically plausible implementation of Bayesian classifiers based on nonparametric kernel-based probability density estimators. The weights of the network store a model of background, which is continuously updated. The PDF estimates consist of a fixed number of kernels, which have fixed width. Results superior to those of Li *et al.* and MoG with 30 Gaussians are reported in [5]. The BNNs attempt to address the problem of computational complexity of the kernel-based background models by exploiting the parallelism of neural networks.

GPUs are receiving increased attention in the computer vision community[9], due to the sharpest rise in peak performance when compared with CPUs and Field Programable Gate Arrays(FPGAs) of the same generation[7]. A single GPU-based implementation of foreground segmentation in the published literature has been described by Griesser *et al.*[8]. Their approach is targeted for human-computer interfaces in indoor environment and not suitable for segmentation when the background undergoes complex changes, since they use a static background image to represent the background. Rather than creating the probabilistic model, they derive more sophisticated features for pixel regions and rely on a collinearity criterion to achieve segmentation. Here we describe an implementation of a state-of-the-art probabilistic model, capable of dealing with both indoor and outdoor sequences with complex motion in the background.

3 Background-modelling Neural Network (BNN)

Background Modelling Neural Network (BNN) is a neural network classifier designed specifically for foreground segmentation in video sequences. The network is an unsupervised learner. It collects statistics related to the dynamic processes of pixel-feature-value changes. The learnt statistics are used to classify a pixel as pertinent to a foreground or background object in each frame of the sequence. A detailed discussion of BNN approach, its segmentation performance and applicability can be found in [5].

The BNN classifier strives to estimate the probability of a pixel value X occurring at the same time as the event of a background or foreground object being located at that particular pixel.

In the structure of BNN, shown in Figure 1, three distinct subnets can be identified. The classification subnet is a cen-

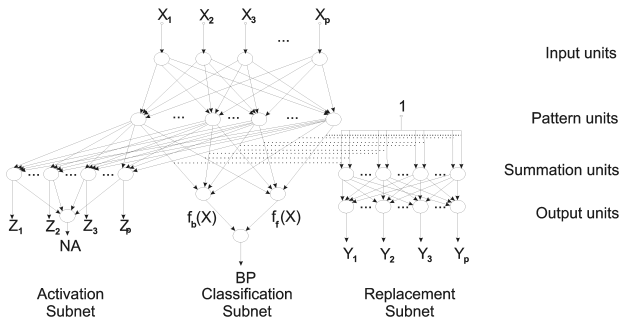


Figure 1. Structure of Background Modeling Neural Network.

tral part of BNN concerned with approximating the PDF of pixel feature values belonging to background/foreground. It is a neural network implementation of a Parzen (kernel based) estimator [12]. Weights between the pattern and summation neurons of the classification subnet are used to store the confidence with which a pattern belongs to the background/foreground. The weights of these connections are updated with each new pixel value received (i.e. with each frame). The output of the classification subnet indicates whether it is more probable that the input feature value is due to a background object rather than a foreground object.

The activation subnet is designed to determine which of the neurons of the network has the maximum activation (output) and whether that value exceeds a threshold (θ) provided as a parameter to the algorithm. If it does not, the BNN is considered inactive and replacement of a pattern neuron's weights with the values of the current input vector is required. If this is the case, the feature is considered to belong to a foreground object. The implementation described in Section 4, uses an iterative approach to determine the maximum output value of a pattern neuron and the index of the pattern neuron that generated the output, rather than implementing the subnet itself. The current maximum value is compared with the output of pattern neurons as they are evaluated and replaced if need be. Once the maximum output is determined it is compared with the threshold to determine if the network is active. This simplified the implementation without affecting performance.

The function of the replacement net is to determine the pattern neuron that minimizes the replacement criterion. The criterion is a mathematical expression of the idea that those patterns that are least likely to belong to the background and those that provide least confidence to make the decision should be eliminated from the model. Again, the procedure is implemented as an iterative emulation of a WTA network originally proposed for this purpose in [5]. To form a complete background-subtraction solution a single instance of a BNN is used to model the features at each pixel of the image.

4 BNNs on a GPU

Our implementation of the BNN approach on a GPU relies on NVIDIA's Compute Unified Device Architecture (CUDA). CUDA is a parallel programming model and software environment designed to overcome the challenge of developing application software that transparently scales its parallelism to leverage the increasing number of GPU processor cores, while maintaining a low learning curve for programmers familiar with standard programming languages such as C [1]. A compiled CUDA program can execute on any number of processor cores, and only the runtime system needs to know the physical processor count.

CUDA extends C by allowing the programmer to define C functions, called kernels, that, when called, are executed N times in parallel by N different CUDA threads. Due to the limited number of registers available to each thread, care should be given to keep the kernel code simple and avoid control flow instructions. The threads are organized in blocks. Threads within a block can cooperate among themselves by sharing data through some shared memory and synchronizing their execution to coordinate memory accesses. Blocks have to be designed so that they can be executed independently and in arbitrary order. Using a high number of threads per block has the advantage of hiding the latency of memory accesses to achieve maximum occupation of the multiprocessor computational units. The number of threads per block is limited by hardware capabilities.

A typical flow of a program using CUDA involves starting the CPU part of the application on the host computer, which then transfers the input data to GPU device memory and invokes a kernel. Once the kernel completes the execution the data can be transferred from the device memory back to the host memory. To minimize memory access latency, the data stored in device memory, should be suitably aligned and preferably accessed as successive 32 bit words.

The BNN approach described in Section 3 allows for computations to be parallelized at the level of single pattern neurons. Fig. 2 shows the program flow of the GPU implementation for a single input sequence frame. The CUDA kernels are listed on the right-hand side of the figure, in the shaded rectangle. The CPU subroutines of the algorithm are listed on the left-hand side. The frames of the video are read by the CPU. The response of each of the patterns stored in the PNN is evaluated separately on the GPU, the summation neuron state incremented and the the function of the WTA network emulated by comparing the maximum value determined up to that point with the response of the pattern neuron evaluated. Once all the pattern-neuron responses

have been evaluated, foreground segmentation is performed by determining which pixel-related BNNs show low activity and evaluating the output for active BNNs. After the segmentation, the weights of the PNN subnets are updated and replacement criteria evaluated for each pattern neuron. Finally, required pattern neurons are replaced by the CPU (host). Probabilistic background subtraction usually relies on postprocessing to remove single pixel effects and small components from the segmentation mask. This step is also performed by the CPU. The postprocessing used, relied on removing connected components smaller than an expected object size.

We experimented with two different setups with respect to memory usage. In the first the BNNs were primarily stored in the host memory and pattern weights for each pattern copied to the device memory when needed and retrieved afterwards. In the second, all the BNN data was stored in the GPU, reducing the number of data transfers between the GPU and the host, but increasing the GPU memory requirements. The data transfers eliminated are indicated in color in Fig. 2.

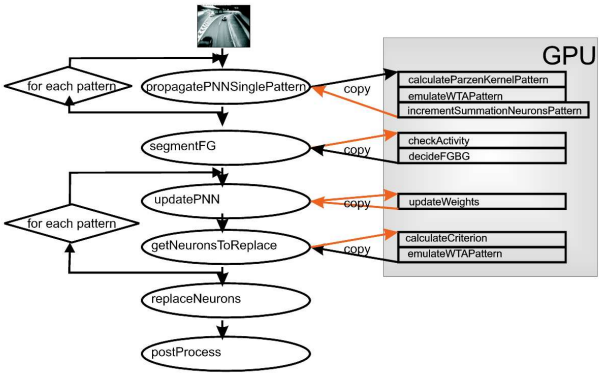


Figure 2. Program flow of GPU implementation.

5 Results

In addition to CUDA SDK, we used OpenCV for video input and output, as well as the postprocessing of the results. The code is available at `http://www.dubravkoculibrk.org/BNN` for research purposes.

We evaluated the algorithm using a large number (30+) of publicly available sequences related to surveillance. The data included complex-background data sets used in [5][10][11][4], as well real-world surveillance sequences collected by our research group. Since the GPU implementation does not affect the segmentation accuracy, we focus on computational performance. Nevertheless, Fig. 3 shows sample segmentation frames obtained for several sequences. Images in the middle correspond to the output of the neural networks. Green pixels correspond to values that were already in the model, but were classified as foreground. The yellow pixels correspond to inactive BNNs, i.e. pixels not currently in the background model. Post processing was performed using connected-components analysis available within OpenCV. The holes in the foreground objects were filled and objects with bounding-box area smaller than 200 pixels were removed.

Tests were run on two different NVIDIA GPUs: Somewhat outdated GeForce 8400M GS designed for portable computers, which has 16 CUDA cores, 256 MB of memory and a core clock of 400 MHz and GeForce GTS 250, which has 128 CUDA cores, 1 GB of memory and a core clock of 738 MHz. Table 1 shows the running times obtained for GeForce 8400M on sequences with commonly used resolutions, for both the approach based on the BNNs stored in GPU (A time) and host memory (B time). The running times were obtained for BNNs containing 10 pattern neurons, running 256 threads on 8400M and 384 threads on 250 GTS. Learning rate (β) was set to 0.05, the activation threshold (θ) was set to 0.5 and the smoothing parameter (σ) to 7. The numbers shown do not include postprocessing, but include the time needed to do data transfers between the host and the device. Frame rates of 10-15 frames per second (67-100 ms/frame) on 160×120 frames are usually considered enough for real-time processing complex-background algorithms [5][14][10]. As Table 1 indicates, BNN approach, when implemented on GPU is able to achieve real-time segmentation of Standard Definition (SD) television sequences. In the case of 160×128 resolution sequences, such as "Curtain", it achieved a processing time below **1** ms (averaged over 10 runs). This represents an order of magnitude speedup (environ 60 times) wrt. results reported by Li *et al.* for their approach. The speed of the application is in this case dominated by the frame input and output operations, rather than the processing. The elimination of additional data transfers between the host and device resulted in average speedup of 1.38 times for 8400M GS and 1.51 times for 250 GTS, in general.

6 Conclusion

In the paper a GPU-based implementation of a state-of-the-art probabilistic foreground segmentation algorithm is described using CUDA and OpenCV. The implementation enables segmentation to be performed in the presence of complex backgrounds at frame rates and resolutions never reported before for any probabilistic approach, as well as making the BNN approach able to achieve segmentation in real-time. BNN-based foreground segmentation, running on commercial programmable graphics hardware, is capa-

Test sequence	Resolution	Frames	A time (ms/frame)		B time (ms/frame)	
			8400M	250 GTS	8400M	250 GTS
"Curtain"	160x128 (QCIF)	2960	33	< 1	54	16
"Water"	320x240	203	93	15.7	130	31
"Container"	352x288(CIF)	300	120	31	165	47
"Bridge"	384x288	2250	130	24	180	47
"Tunnel"	720x576 (SD)	9018	N/A	**95**	542	145

Table 1. Running times for NVIDIA GeForce 8400M GS and 250 GTS: A-BNNs stored on device, B-BNNs stored on host.

ble of handling Standard Definition (SD) television video in real-time and can process 160×128 resolution sequences at speeds dominated by the time it takes to load video frames and transfer them to the GPU.

In addition to addressing several practical implementation aspects of the original algorithm, we proposed two different strategies for storing the neural network data and compared the results on a set of commonly used surveillance videos. Future work will address the issues of moving the processing completely to the GPU, including the post-processing and to GPU- and neural-network-based tracking, as a logical extension of the proposed methodology.

References

[1] Nvidia cuda programming guide version 2.0. `http://www.nvidia.com/object/cuda_develop.html`, Dec. 2008.

[2] T. Boult, R. Micheals, X.Gao, P. Lewis, C. Power, W. Yin, and A. Erkan. Frame-rate omnidirectional surveillance and tracking of camouflaged and occluded targets. In *Proc. of IEEE Workshop on Visual Surveillance, pp. 48-55*, 1999.

[3] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.

[4] D. Culibrk, B. Antic, and V. Crnojevic. Real-time stable texture regions extraction for motion-based object segmentation. In *Proceedings of the 20th British Machine Vision Conference (BMVC 2009)*, Sept. 2009.

[5] D. Culibrk, O. Marques, D. Socek, H. Kalva, and B. Furht. Neural network approach to background modeling for video object segmentation. In *IEEE Trans. on Neural Networks, vol. 18*, number 6, pages 1614 – 1627. IEEE Press, Nov. 2007.

[6] A. ElGammal, R. Duraiswami, D. Harwood, and L. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. In *Proc. of the IEEE, vol. 90, No. 7, pp. 1151-1163*, 2002.

[7] M. Gipp, G. Marcus, N. Harder, A. Suratanee, K. Rohr, R. Knig, and R. Mnner. Haralicks texture features computed by gpus for biological applications. *IAENG International Journal of Computer Science*, 36(1), Feb. 2009.

[8] A. Griesser, S. D. Roeck, A. Neubeck, and L. V. Gool. Gpu–based foreground–background segmentation using an extended colinearity criterion. In *Vision, Modeling, and Visualization (VMV 2005)*, Nov. 2005.

[9] S. Lefebvre, S. Hornus, and F. Neyret. *GPU Gems 2 - Programming Techniques for High-Performance Graphics and General-Purpose Computation*, chapter Computer Vision on the GPU, pages 651–667. Addison Wesley, 2005.

[10] L. Li, W. Huang, I. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. In *IEEE Trans. Image Processing, vol. 13, pp. 1459-1472*, 2004.

[11] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh. Background modeling and subtraction of dynamic scenes. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 1305–1312, Washington, DC, USA, 2003. IEEE Computer Society.

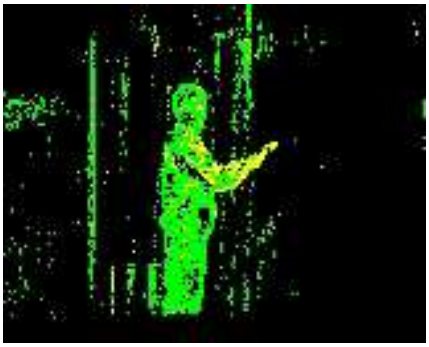
[12] E. Parzen. On estimation of a probability density function and mode. In *Ann. Math. Stat., Vol. 33, pp. 1065-1076*, 1962.

[13] Y. Sheikh and M. Shah. Bayesian modeling of dynamic scenes for object detection. In *IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 27, pp. 1778-1792*, 2005.

[14] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. In *IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, pp. 747-757*, 2000.

[15] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *International Conference on Pattern Recognition (ICPR)*, volume 2, pages 28–31, Aug. 2004.

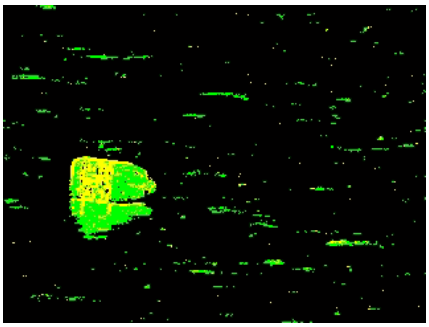
(a) ”Curtain” sequence frame.



(b) Segmentation.

(c) Post-processed.

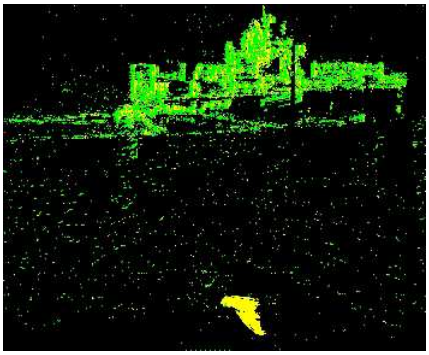
(d) ”Container” sequence frame.



(e) Segmentation.

(f) Post-processed.

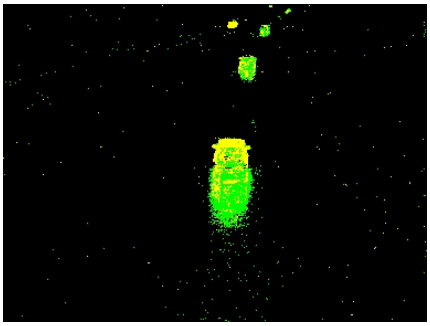
(g) ”Container” sequence frame.



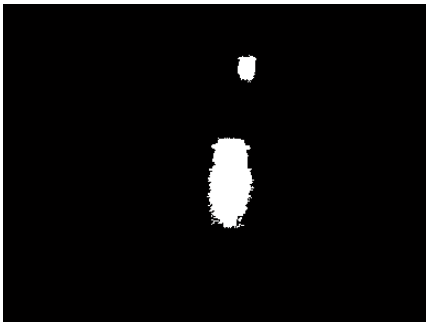
(h) Segmentation.

(i) Post-processed.

(j) ”Bridge” sequence frame.

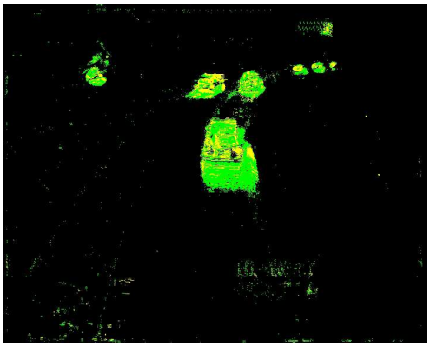


(k) Segmentation.



(l) Post-processed.

(m) ”Tunnel” sequence frame.



(n) Segmentation.

(p) Post-processed.

Figure 3. Segmentation results for sample frames.