2010 International Symposium on Nonlinear Theory and its Applications
NOLTA2010, Krakow, Poland, September 5-8, 2010

GPGPU Accelerated Scene Segmentation Using Nonparametric Clustering

Balázs Varga and Kristóf Karacs

Dept. of Information Technology, Pázmány Péter Catholic University
Práter str. 50/a, H-1083, Budapest, Hungary
E-mail: varga.balazs@itk.ppke.hu, karacs@itk.ppke.hu

Abstract– This paper presents the parallel implementation of a nonparametric image segmentation method: the mean shift algorithm using joint spatial-range feature space. By considering spatial information, the mean shift can distinguish topographically differing objects in the scene, but this feature costs additional computational demand through increased number of kernel functions. The proposed algorithm runs the mode-defining kernel iterations parallel by utilizing the many-core processor architecture present in the general-purpose graphics processing unit (GPGPU). We use our own voting procedure for pixel-cluster assignment. Numerical evaluation showed that our solution efficiently speeds up the image clusterization procedure.

1. Introduction

Scene segmentation is one of the most common, yet most versatile tasks of image processing. Among today's most advanced segmentation approaches, such as graph cuts [1][2], normalized cuts [3], or various types of k-means [4][5], the mean shift segmentation is one of the most studied and applied nonlinear techniques. Basically having no *a-priori* knowledge demand, it can dynamically set the number of segmented clusters through a nonparametric framework. Our solution gives a notable speed-up to the mean shift method, by parallelizing its internal structure, and instead of running it on a CPU unit, we use a many-core programmable *general-purpose graphics programming unit* (GPGPU) [6]. For now, we did not aim to overcome the quality of the original algorithm implemented on the CPU, rather to show that our GPGPU-based implementation can achieve similar segmentation accuracy with considerably lower time demand.

The paper is organized as follows: in the first part of the second section we give an overview of the generic, joint feature space mean shift paradigm. The second part of the section briefly summarizes related works about possible speed-up strategies; while the third part describes the disadvantage behind the usage of the spatial domain. The third section explains our solution of speeding up this task. Section four presents the evaluation framework, which is followed by the numerical results in section five. The paper is closed with a short summary in the sixth section.

2. Algorithmic background

After the mean shift procedure was introduced by Fukunaga and Hostetler [7] in 1975, it was Cheng [8] who pointed out 20 years later that the mode seeking process of the algorithm is a parallel hill climbing method, applying the clusterization algorithm in the Hough space. Following another half a dozen of years of smoldering, Meer and Comaniciu gave an extensive overview [9] of the segmentation framework, using it for image segmentation and discontinuity preserving smoothing. Their approach concerning color images is briefly summarized in subsection 2.1.

2.1 Mean shift in the joint feature space

The mean shift procedure considers its feature space as an empirical probability density function. A local maximum of this function (namely, a region over which it is highly populated) is called a mode. Mode calculation is formulated as an iterative scheme of mean calculation, which takes a certain number of feature points and calculates their mean value by using a weight kernel function. Meer and Comaniciu used a composite feature space consisting of both topographical (*spatial*) and color (*range*) information of the image. As a result, each feature point in this space is represented by an $\boldsymbol{\chi}=(x_r;x_s)$ 5D vector which consists of the corresponding pixel's $x_s=(x,y)$ 2D position in the spatial lattice, and its x_r 3D color value in the applied color space (e.g., the $x_r=(Y,Cb,Cr)$ coordinates). The iterative scheme for the calculation of a mode is as follows: let $\boldsymbol{\chi_i}$ and $\boldsymbol{z_i}$ be the 5D input and output points in the joint feature space for all i=[1,n], n being the number of pixels in color image I.

Then for each i

1. Initialize $k = 1$
2. Compute the iterative formula

$$\chi_i^{k+1}=\frac{\sum_{j=1}^{n}\chi_j\, g\left(\left\|\frac{x_{r,j}-x_{r,i}^k}{h_r}\right\|^2\right) g\left(\left\|\frac{x_{s,j}-x_{s,i}^k}{h_s}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x_{r,j}-x_{r,i}^k}{h_r}\right\|^2\right) g\left(\left\|\frac{x_{s,j}-x_{s,i}^k}{h_s}\right\|^2\right)} \qquad (1)$$

until the mean shift vector $\left\| \chi_i^{k+1} - \chi_i^k \right\|$ falls under a given threshold, where *g* denotes the Gaussian kernel function, with h_s and h_r being the spatial-, and range bandwidth parameters respectively.

3. Allocate $z_i = \chi_i^{k+1}$; that is, output value $\boldsymbol{z_i}$ is given by feature point $\boldsymbol{\chi_i}$ after the final $(k+1)^{st}$ step.

Those $\boldsymbol{z_i}$ points, which are adequately close to each other, are concatenated resulting discrete, non-overlapping clusterization of the input image. All pixels in the cluster inherit the color of its mode.

The main advantage of applying the joint feature space is that the algorithm became capable of discriminating scene objects based on their color *and* position; making mean shift a multi-purpose, nonlinear, nonparametric tool for image segmentation.

On the other hand the disadvantage of the algorithm, as it was specified earlier by Cheng is its high computational complexity of $O(n^2)$.

2.2 Acceleration strategies

Comaniciu and Meer introduced an efficient technique called the coarse grid [10] in order to highly reduce complexity. Briefly, they perform random tessellation of the feature space with $m << n$ kernels, execute mean shift segmentation (resulting $\boldsymbol{z_i}$, i=[1,m] modes), merge close modes (as in [9]), then assign each $\boldsymbol{\chi_i}$ feature point to the closest $\boldsymbol{z_j}$ cluster-defining mode. They showed that this approach is capable of producing technically equal segmentation quality with a computation demand of $O(m*n) << O(n^2)$.

Ever since, several alternative techniques have been proposed to achieve speed-ups, e.g. through space discretization and downsampling [11], local subsets [12], expectation-maximization [11][13], hierarchical solutions [14][15], and the Newton iteration method [16][17]. Although our current system does not use these alternative techniques, later on most of them can easily be added to our framework enhancing its speed and reducing its time demand.

2.3 Over-segmentation: the advantage's tradeoff

The clear advantage brought by the usage of the spatial domain is the topographical discriminative potential: objects with similar or even the same color can be distinguished, if they are topographically distinct. On the other hand spatial discrimination of two discrete objects requires the usage of two kernels. Therefore in the case of very detailed images, spatial filtering involves a computational tradeoff: proper coverage of the feature space necessitates the usage of numerous kernels.

3. Our approach

As a result, both the original and the improved algorithms follow the bottom-up strategy, when the output is a result of over-segmentation, which is followed (off-line), or accompanied (on-line) by a cluster merging procedure.

We considered the off-line mean shift algorithm, which is divided into two main subtasks: the mode calculation and the cluster merging procedure. Mode calculation algorithm is a highly data parallel [18] task: the same iterative procedure is performed on the elements of the feature space with each kernel having a different seed point. Nowadays many-core GPGPUs are publicly available at a reasonable price, having more than a hundred distinct stream processors, which can effectively handle data-parallelism-related tasks. We implemented the mode seeking task on a GPGPU, and compared its performance with the CPU implementation of the procedure.

3.1 GPGPU implemented mode calculation

Our algorithm applies Comaniciu and Meer's coarse grid technology [10] on the joint feature space. But instead of running the iteration specified by eq. 1 on a single kernel until saturation, we extend this computational framework by running the iterative algorithm simultaneously on several mean shift kernels, which we call *multiple simultaneous mode seeking* (MSMS).

The MSMS begins by selecting *m* initial mean points randomly in the joint feature space. The mean shift iteration is then started from these seed points by using Gaussian kernel functions having a common (h_s,h_r) spatial-range bandwidth parametrization for each kernel. The procedure is terminated, when the length of each kernel's mean shift vector becomes smaller than a pre-defined threshold value.

In order to properly assign all feature points to the corresponding mode, we constructed a voting system. In every iteration of mode seeking, for each kernel we compute pixel-wisely the following *cumulative confidence value* (CCV):

$$C_{i,j}^{k+1} = C_{i,j}^{k} + g\left(\left\|\frac{x_{r,j} - x_{r,i}^{k}}{h_r}\right\|^2\right) g\left(\left\|\frac{x_{s,j} - x_{s,i}^{k}}{h_s}\right\|^2\right) \qquad (2)$$

with

$$C_{i,j}^{0} = 0 \qquad (3)$$

where $C_{i,j}^{k}$ denotes the confidence value of pixel *i* at the k^{th} iteration for kernel *j*. Note that the calculation of the CCV does not require additional computation, as it is a part of the mean shift iteration's numerator. Let $C_{i,j}$ denote the final CCV computed in the last iteration and let CID_i stand for the i^{th} pixel's *cluster ID*. After the modes are retrieved and every $C_{i,j}$ has been obtained, each image

pixel gets associated with a mode using the following rule:

$$CID_i = \arg\max_j (C_{i,j}) \quad (4)$$

3.2 Cluster merging procedure

The number and structure of final clusters are constructed with cluster merging, which currently runs on the CPU. Cluster *i* and *j* are joined, if they satisfy the criteria:

C1. The two clusters have a common border in terms of the eight-neighbor connectivity.

C2. $\left\| x_{r,i} - x_{r,j} \right\| < h_r$.

In this case the position of the mode is recalculated. Let NP_i and NP_j denote the number of pixels in clusters *i* and *j* respectively, and let us say that both C1 and C2 criteria holds and the two are merged into cluster *k*. Then the color information carried by mode z_k of the newly formed cluster is

$$z_k = \frac{NP_i * z_i + NP_j * z_j}{NP_i + NP_j} \quad (5)$$

i.e. it is a weighted average of the conjoined duet. The procedure runs iteratively until no classes can be merged.

4. Evaluation framework

To provide comparative results to the proposed GPGPU algorithm, the coarse grid CPU-optimized mean shift procedure was implemented with as few differences from the GPGPU version as possible. The main difference is that while the GPGPU runs the MSMS version, the CPU does the mode calculation one by one (*single simultaneous mode seeking*, SSMS).

Image segmentation is an ill-posed task [19], but several publicly available image corpora exist to make algorithms comparable. For evaluation purposes, we selected 50 color images from the Berkeley Segmentation Dataset (BSDS), [20] for which multiple human-made segmentation maps are provided as reference. Let the name *best parameter pair* (BPP) denote a pair of (h_s, h_r) kernel bandwidth parameters that result in a closest-to-the-reference segmentation for a given image of our *evaluation image set*.

In order to determine such pairs, the CPU algorithm was utilized the following way: 64 alternative segmentations were made for each evaluation image using 8 x 8 different (h_s, h_r) bandwidth values for each of the 50 images. During the process, the algorithm computed and logged the number of clusters before and after the merging procedure, and the elapsed time of the mode seeking. Then, depending on the number of provided BSDS references 1 to 4 BPPs were selected for each image, resulting a total of 117 pairs. Next, the GPGPU algorithm was run on the evaluation set using the corresponding BPPs, and finally the elapsed time of the mode seeking was compared to the CPU algorithm. It is worth to note that the GPGPU runtime includes the time-demanding CPU to GPGPU and GPGPU to CPU data transfers. Furthermore, since the number of mean shift iterations depends on the (randomly selected) initial kernel position, we ran both algorithms multiple times with the same parameterization, and compared the average running times. Moreover cluster merging was done using the same CPU-based algorithm; therefore time consumption of the mode merging procedure was not part of the comparison.

All calculations were run on a single PC equipped with 2GB of RAM and an Intel E6400 CPU running at 2.13GHz. The GPGPU was an nVidia G92 GPU operating with 112 stream processors and 1024MB of video RAM.

5. Results

On average, the GPGPU was able to segment the scene 2.997 times faster under the same circumstances. Fig. 1/a displays an example image from the BSDS among with a segmentation made by human observer on 1/d, and our segmentation results made on the different computational platforms on 1/b and 1/c.

6. Conclusion

We successfully implemented the nonparametric mean shift clustering algorithm using the joint spatial-range feature space onto the many-core GPGPU, for which we used our own voting procedure for mode selection. The GPGPU algorithm proved to run almost three times faster than its CPU variant. Later on we plan to enhance the system using the acceleration strategies described in Section 2.2.

References

[1] V. Kwatra, A. Schodl, I. Essa, G. Turk, A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Transactions on Graphics*, vol. 22, 2003, pp. 277–286.

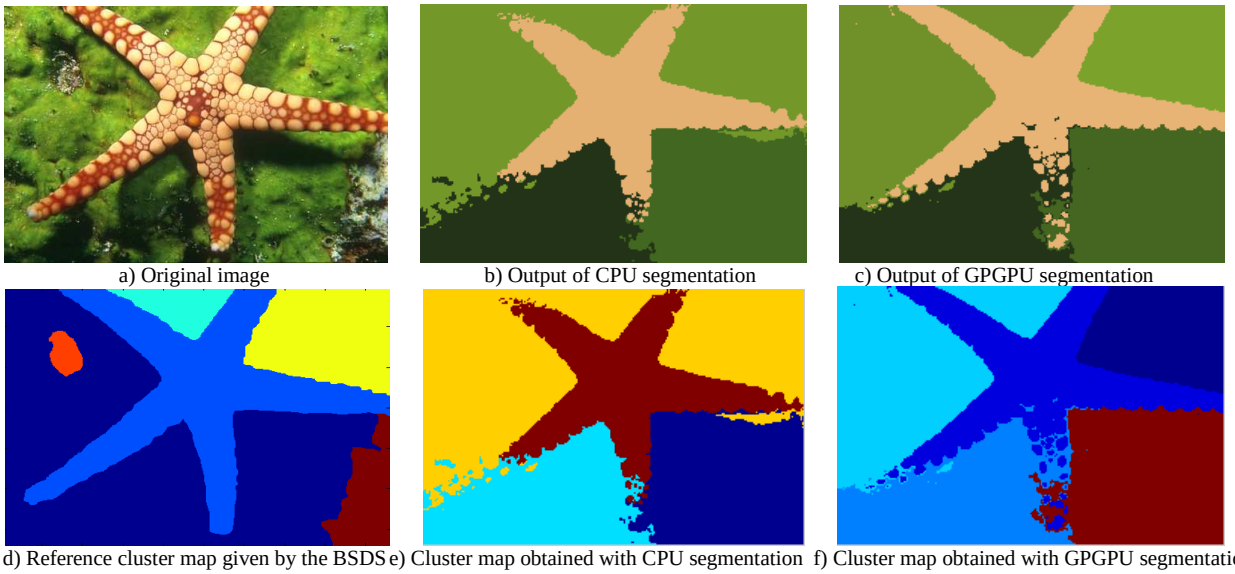
[2] P. Felzenszwalb D. Huttenlocher. "Efficient graph-based segmentation algorithm," *IJCV*, 2004.

[3] J. Shi and J. Malik. "Normalized cuts and image segmentation," *IEEE Transactions* on *Pattern Analysis* and *Machine Intelligence*, 22(8):888–905, 2000.

[4] P.S. Bradley U.M. Fayyad, "Refining initial points for k-means clustering," *Proc. 15th International Conference on Machine Learning*, 1998.

[5] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, A.Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transactions* on *Pattern Analysis* and *Machine Intelligence*, vol. 24, 2002, pp. 881–892.

[6] D. Luebke, M. Harris, N. Govindaraju, A. Lefohn, M. Houston, J. Owens, M. Segal, M. Papakipos, I. Buck, "GPGPU: general-purpose computation on graphics hardware," *Proc. of the 2006 ACM/IEEE Conference on Supercomputing*, Tampa, Florida: ACM, 2006, pp. 208.



a) Original image b) Output of CPU segmentation c) Output of GPGPU segmentation

d) Reference cluster map given by the BSDS e) Cluster map obtained with CPU segmentation f) Cluster map obtained with GPGPU segmentation

Figure 1. An example of segmentation made on the different computation platforms. Segmented images are result of mode seeking followed by cluster merge. $(h_s, h_r) = (0.08,0.01)$, $m = 54$ clusters were merged into 4 (CPU case) and 5 (GPGPU case)

[7] K. Fukunaga, L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions* on *Information Theory*, vol. 21, pp. 32–40, 1975.

[8] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions* on *Pattern Analysis* and *Machine Intelligence*, vol. 17, pp. 790–799, 1995.

[9] D. Comaniciu, P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions* on *Pattern Analysis* and *Machine Intelligence*, vol. 24, 2002, pp. 603–619.

[10] D. Comaniciu, P. Meer, "Distribution free decomposition of multivariate data," *Pattern Analysis and Applications*, vol. 2, pp. 22–30, 1999.

[11] M.A. Carreira-Perpinan, "Acceleration strategies for Gaussian mean-shift image segmentation," *CVPR*, 2006.

[12] K. Zhang, M. Tang, J.T. Kwok, "Applying neighborhood consistency for fast clustering and kernel density estimation," *Proc. CVPR* 2005, pp. 1001 – 1007, 2005.

[13] M.A. Carreira-Perpinan, "Gaussian Mean-Shift is an EM algorithm," *IEEE Transactions* on *Pattern Analysis* and *Machine Intelligence*, vol. 29, 2007, pp. 767-776.

[14] S. Paris and F. Durand, "A topological approach to hierarchical segmentation using mean shift," Proc. *CVPR*, 2007.

[15] D. DeMenthon. "Spatio-temporal segmentation of video by hierarchical mean shift analysis," *Statistical Methods in Video Processing Workshop*, 2002.

[16] M.A Carreira-Perpinan, "Mode-finding for mixtures of Gaussian distributions," *IEEE Transactions* on *Pattern Analysis* and *Machine Intelligence*, 22(11) pp. 1318–1323, November 2000.

[17] C. Yang, R. Duraiswami, D. DeMenthon, L. Davis, "Mean-shift analysis using quasi-Newton methods," *Int. Conf. on Image Processing*, 2003.

[18] W.D. Hillis, J. Guy, L. Steele, "Data parallel algorithms," *Communications of the ACM*, vol. 29, 1986, pp. 1170-1183.

[19] T. Poggio, V. Torre, "Ill-posed problems and regularizaron analysis in early vision," *Artificial Intelligence Lab.* Memo, vol. 773.

[20] D. Martin, C. Fowlkes, D. Tal, J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," *Proceedings of the Eighth IEEE International Conference on Computer Vision. ICCV 2001*, Vancouver, BC, Canada, pp. 416-423.