# GPU BASED FAST MPEG-CDVS ENCODER

*Wei Sun* [1]*, Xinfeng Zhang*[2]*, Shiqi Wang*[3]*, Jie Chen*[1]*, Ling-Yu Duan*[1]

[1]Institute of Digital Media, Peking University, Beijing, China
[2]Rapid-Rich Object Search Laboratory, Nanyang Technological University, Singapore
[3]Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong
{weisun199508, lingyu, cjie}@pku.edu.cn, xfzhang@ntu.edu.sg, shiqwang@cityu.edu.hk

## ABSTRACT

The compact descriptors for visual search (CDVS) standard from ISO/IEC Moving Picture Experts Group (MPEG) has received increasing attentions due to its effectiveness in mobile visual search related applications. To explore the efficiency of CDVS in real-time applications, we implement the first optimized CDVS encoder based on the graphics processing unit (GPU). In particular, the CDVS feature extraction is implemented in a CPU and GPU collaborative architecture, and the most of the computation-intensive operations are transferred to GPU platform, which achieves significant speedup for CDVS compared with CPU implementation. Extensive evaluations on standard datasets provide promising results of the proposed scheme for real applications scenarios.

***Index Terms***— MPEG-CDVS, GPU, feature extraction

## 1. INTRODUCTION

Recently, there has been an exponential increase in the demand for mobile visual search [1], which can facilitate many potential applications such as product identification, landmark localization, augmented reality, etc. To facilite these services, the Moving Picture Experts Group (MPEG) has published the Compact Descriptors for Visual Search (CDVS) standard in 2015 [2, 3]. The MPEG-CDVS standard provides the standardized bitstream syntax to enable interoperability for visual search. The normative blocks of CDVS mainly include the local feature descriptor extraction and aggregation, such that handcrafted local and global feature descriptors are compactly represented at different descriptor lengths to achieve bit-rate scalability (e.g., 512B, 1KB, 2KB, 4KB, 8KB, and 16KB).

The MPEG-CDVS is comprised of two kinds of features, *i.e.,* local feature and global feature. For local features, it includes the modules of interest point detection, local feature selection, description and compression [2]. Specially, the low-degree polynomial [4] is adopted in MPEG-CDVS to detect the interest points, and then the popular scale-invariant feature transform (SIFT) descriptors are constructed and subsequently compressed by transform and scalar quantization. For global features, the selected SIFT descriptors are aggregated to the Fisher Vectors (FV), which can be further binarized for fast comparison. To compress the global feature descriptor, a subset of gradient vectors is selected based on the rankings in terms of their individual standard deviation [5].

In addition to the bandwidth consumption, the extraction speed of CDVS descriptors directly determines the visual search latency and interactive experience. Therefore, highly efficient CDVS implementation and optimization strategies are strongly desired in the practical applications of CDVS. In particular, with the popularity of large scale video analysis, real time CDVS feature extraction of video frames should also be enabled to support video analysis applications with compact descriptors such as mobile augmented reality, automotive, surveillance, media entertainment, etc [6]. Moreover, recently it has been observed that there are complementary effects of CDVS and deep learning based features, and the combination of both approaches can obtain the state-of-the-art performance in image and video analysis [7, 8]. As deep learning based features can be efficiently extracted by the graphics processing unit (GPU) [9, 10], how to leverage such infrastructure and achieve high efficiency CDVS feature extraction with GPU is also worth investigated. These emerging requirements all pose new challenges to highly efficient extraction of features standardized by MPEG-CDVS.

Recently, the popularity of GPU [11], which provides the parallel-processing capabilities for highly data parallel applications, has enabled many fast feature extraction applications including SIFT [12–16], speeded-up robust features (SURF) [17], FV [18] as well as deep learnt features [9, 10]. In order to further push CDVS towards industry applications, this paper provides a practical strategy for GPU based CDVS implementation and a detailed analysis for efficiency of GPU-based CDVS. The main contribution of this paper is that we design the first practical GPU based CDVS implementation framework, which achieves significant speedup compared with traditional CPU based encoders. The CDVS fea-

Ling-Yu Duan is the corresponding author

ture extraction speed at about 3 ms/pic can be achieved on the NVIDIA Quadro GP100 graphics card.

The rest of this paper is organized as follows. In Section 2, we present the implementation of the CDVS encoder based on GPU. Experimental results are provided in Section 3, and finally we conclude this paper in Section 4.

## 2. THE FRAMEWORK OF GPU BASED CDVS

In this paper, we design the first CDVS encoder based on GPU platform to speed it up, which is the fastest implementation of CDVS at present to our best knowledge. Considering the computational complexity and parallelism on GPU, we propose a heterogeneous GPU-CPU framework as illustrated in Fig. 1, which will be introduced in section 2.3.

### 2.1. Overview of CDVS

The CDVS mainly includes five computation-intensive components, *i.e.*, interest point detection, selection, description, compression and aggregation. In interest point detection, an image pyramid is constructed by a series of Gaussian filters with different scale parameters and a low-degree polynomial (ALP) LoG detector is applied to find the interest points. Different from existing interest point detection, CDVS adopted a more efficient feature selection strategy by sorting the interest point candidates according to image statistical characteristics, and only a subset of points are selected. Similar with SIFT features, CDVS also utilizes local orientation histograms in a neighborhood centered the interest points to construct the local feature descriptors. Besides that, a high efficient and compact global feature representation for each image is constructed by aggregating the uncompressed local descriptors. Finally, the transformed local descriptors as well as its corresponding locations are compressed via entropy coding, which constitutes the CDVS bitstream jointly with global feature.

### 2.2. Computation Analysis for CDVS on CPU and GPU

To achieve high efficient implementation of CDVS, the parallel-friendly modules with intensive computations can be transferred to the GPU platform, while the modules with significant data dependency are processed in CPU.

#### *2.2.1. Computational complexity analysis of CDVS*

To analyze computational complexity of CDVS encoder on CPU, we measure the average running time of each module using TM14.0 on 1000 images with resolution $640 \times 480$. Fig. 2 shows the percentage of average running time for different modules, and for images with common resolutions, we can regard the interest point detection, local feature description and aggregation as time-consuming components, which take up more than 99.5% running time for the CDVS encoder.

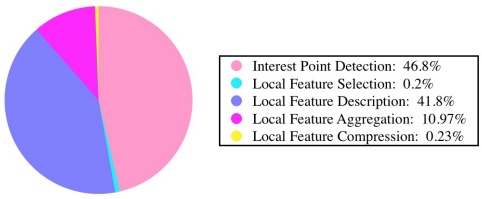


**Fig. 2**. The running time distribution for main modules in CDVS, tested on a Linux PC with Intel(R) Xeon(R) CPU E5-2650 v2@2.60GHz.

#### *2.2.2. Parallelism analysis on GPU*

The architecture of GPU is different from CPU, the computations of which are controlled by the same processing unit. GPU possesses hundreds even thousands of cores to execute the same instruction simultaneously. Therefore, GPU can launch with Single Instruction Multiple Threads (SIMT) to execute the same operation to process huge amounts of pixels in parallel. In addition, there is also no branch prediction and data forwarding in GPU. All these features make GPU suitable for computation-intensive, highly parallel data processing instead of data caching and flow control. Therefore, the sorting operation in local feature selection and entropy coding in local feature compression with significant data dependency are suitable for CPU processing, while others can be transferred to GPU platform with high efficiency.

The interest point detection is the most time-consuming module in CDVS, which is implemented by applying convolution on the input image with separable spatial Gaussian filters and extrema detection on the filtered images. The value of each pixel in scale space $(x, y, \sigma)$ is derived from the convolution between Gaussian filter and a contiguous area of input image. In the extrema detection, each pixel in scale space makes comparisons with 26 neighbors. Therefore, these calculations can be divided into the same pixel-wise operations without data dependence.

The local feature description is another computation-intensive module, which mainly calculates the histogram for each interest point in a neighborhood centered the interest point. The gradients of each pixel in the neighborhood are computed and added into histogram. Thus, the operations for each pixel only preserve little data dependence in histogram cumulation. However, these related operations for pixels in the same neighborhood can also be implemented in parallel based on our design.

The PCA and the GMM selection for fisher vector are the main calculations in local feature aggregation. Fortunately, the matrix operations for PCA is well supported on GPU with existing libraries. Moreover, the GMM selection for fisher vector can be divided into a series of matrix calculations between each descriptor and each GMM component.

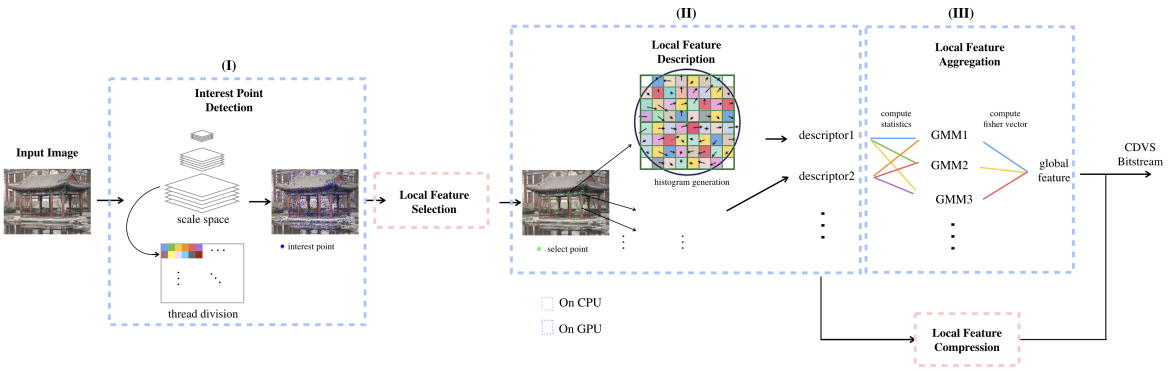


**Fig. 1**. Overview of the proposed heterogeneous GPU-CPU framework of CDVS encoder.

### 2.3. Implementation of CDVS on GPU

Based on the analysis, we transfer the interest point detection, local feature description and aggregation on GPU to speed up CDVS. In interest point detection, we use one thread to each pixel in scale space and check whether it is extremum, as illustrated in Fig. 1 (I) different color blocks representing different threads. The challenge to implement the separable spatial convolution efficiently on the GPU is how to efficiently reduce the memory operations. This is because an arithmetic operation is much faster than a memory operation, *e.g.*, according to [19], arithmetic operation costs 4 cycles per warp on NVIDIA G80, while memory operation costs 400 cycles. In our implementation, we optimize the memory access by putting the data required by threads in one thread block in a shared memory, which is a cache for all the threads in the same thread block. For the extremum detection, the similar optimized scheme proposed in [20] is utilized. To decrease the memory access, instead of comparing with 26 neighboring pixels for each interest point, we firstly perform extreme detection in 8 neighborhood in current scale image, and only when it is an extremum, we further perform the extreme detection in other 18 neighbors of adjacent scale images.

For local feature description, although the feature-wise parallelization of local feature description can be easily implemented, where one thread is assigned to one interest point and independent of each other threads, it will lead to a very unbalanced workload among threads and degenerate the processing efficiency on GPU due to variant neighborhood sizes for interest points. Besides, the local feature selection in CDVS only keeps less than $N$ ($N$ is usually less than 600) features for an image. It means that at most $N$ threads can be launched simultaneously, which is much fewer than cores in GPU. To utilize the computation resource in GPU fully, we design a pixel-wise parallelization in this paper, as illustrated in Fig. 1 (II). For each region around interest point, we utilize the same amount of threads as the number of pixels of the region to compute their gradient magnitudes and orientations in parallel, and then add them into histogram automatically. Histogram can be kept entirely in shared memory due to its relatively small size.

For local feature aggregation, based on a nice CUDA implementation for matrix computation provided in [18], we speed up it by transforming the main computation into matrix operations to process the fisher vector calculation. Fig. 1 (III) shows the thread division, while different colored lines means different threads. For example, we transform the distance calculation illustrated in Eq.(1) between GMM components and compressed descriptors into matrix multiplication and a set of element-wise operations as illustrated in Eq.(2).

$$P_{i,j} = \sum_{k=0}^{31}((D_{i,k} - M_{j,k})^2/V_{j,k}), \tag{1}$$

$$P = D^2 * (1/V^{\mathrm{T}}) - 2D * (M^{\mathrm{T}}/V^{\mathrm{T}}) + O * ((M^{\mathrm{T}})^2/V^{\mathrm{T}}). \tag{2}$$

Here, $P_{i,j}$ denotes the probability of the $i^{th}$ descriptor under the $j^{th}$ GMM component, $D_{i,k}$ denotes the $k^{th}$ dimension of the $i^{th}$ descriptor, $M_{j,k}$ and $V_{j,k}$ denote the $k^{th}$ dimension of mean and covariance vector for the $j^{th}$ GMM component.$O$ is a matrix be filled by ones with the same dimension of $D$. $*$ denotes matrix multiplication while exponent and division calculations are element-wise operations. The same optimization can be used in the generation of a fisher vector.

## 3. EXPERIMENTAL RESULTS

**Experiment configuration.** We evaluate the GPU implementation of CDVS on the MPEG-7 CDVS benchmark dataset [21] [22], which has introduced considerably more varieties in terms of objects, scale, rotation, occlusion and lighting conditions than other popular datasets. In particular, it is composed of 5 classes including graphics, paintings, video frames, landmarks and common objects. Moreover, the images in graphics dataset are further divided into 3 subsets termed as 1a, 1b and 1c, according to image resolution and compression quality.

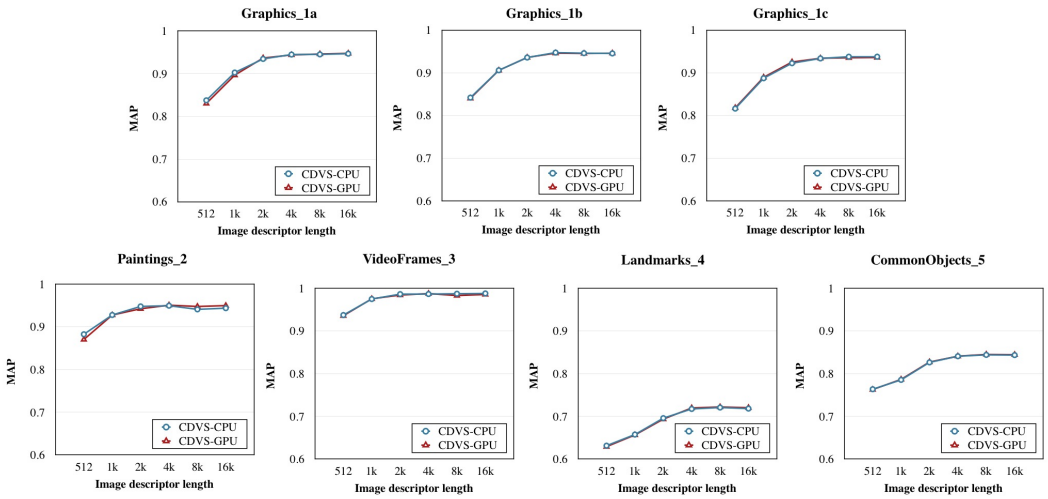


**Fig. 3**. Comparisons of the retrieval performance on the CDVS datasets.

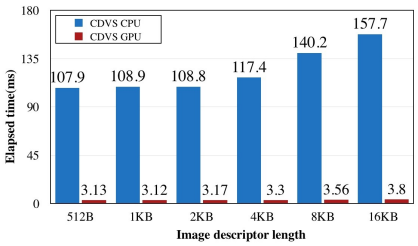


**Fig. 4**. Comparisons of the average running time.

The experiments are conducted using a Linux PC with Intel(R) Xeon(R) CPU E5-2650 v2@2.60GHz and a NVIDIA Quadro GP100 with CUDA 7.5 installed. For the fair comparison, only one core on CPU and one GPU is used in the performance evaluation. The MPEG-CDVS reference software TM14 is utilized as a baseline, denoted by CDVS-CPU, which is the state-of-the-art test model of CDVS encoder as described in [3]. Here, our GPU implementation is denoted by CDVS-GPU.

**Performance evaluation.** We evaluate our GPU implementation on the retrieval performance measured by mAP and computational speed in terms of the execution time. According to CDVS evaluation framework ,a total of 8314 query images, 18840 reference images from CDVS dataset and a distractor set of 1 million images from Flickr [23] are used to evaluate the retrieval performance at 6 pre-defined descriptor lengths: 512 bytes, 1K, 2K, 4K, 8K and 16K. All images are resized into VGA resolution. As shown in Fig. 2.3, comparing with CDVS-CPU in CDVS dataset, CDVS-GPU has achieved almost the same retrieval performance, with about 0.06% mAP loss. The slight drop due to that floating-point errors make some difference on CDVS features bewteen CDVS-CPU and CDVS-GPU, while many fixed parameters in the algorithm that are fine-tuned for CPU version.

Moreover, we provide the computation time comparisons in Fig. 4, which shows the average computational time running on the whole CDVS dataset. As can be seen from Fig. 4, we can obtain a speedup of 37 times on average, and the proposed CDVS encoder can fully utilize the GPU resources in most of the computations due to the highly parallelized design. This provides useful evidence that our GPU implementation can significantly reduce time usage of the CDVS, enabling its practical applications in real scenarios. More experimental results are shown in [24].

## 4. CONCLUSION

We have presented an efficient GPU based MPEG-CDVS encoder. By analyzing the operations of CDVS, we design a practical CDVS implementation strategy, which is fully optimized in terms of the complexity-accuracy performance. Extensive experimental results show that the proposed CDVS encoder achieves significant speedup compared with that on CPU platform, and achieves about 3 ms/pic while maintaining the competitive performance for image retrieval, which is also practical for real-time CDVS based video feature extraction. In future work, we will explore how to harmoniously leverage the merits of highly efficient and low complexity MPEG-CDVS, and the state-of-the-art deep learning based descriptors via GPU or GPU-CPU hybrid computing.

## 5. ACKNOWLEDGMENTS

This work was partially supported by grants from National Natural Science Foundation of China (U1611461, 61661146005) and National Hightech R&D Program of China (2015AA016302).

## 6. REFERENCES


[1] B. Girod, V. Chandrasekhar, D. M. Chen, N.-M. Cheung, R. Grzeszczuk, Y. Reznik, G. Takacs, S. S. Tsai, and R. Vedantham, "Mobile visual search," *IEEE signal processing magazine*, vol. 28, no. 4, pp. 61–76, 2011.

[2] L.-Y. Duan, V. Chandrasekhar, J. Chen, J. Lin, Z. Wang, T. Huang, B. Girod, and W. Gao, "Overview of the MPEG-CDVS standard," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 179–194, 2016.

[3] G. C. Stavros Paschalakis, Gianluca Francini, "Test Model 14: Compact Descriptors for Visual Search," *ISO/IEC JTC1/SC29/WG11/W15372*, 2011.

[4] G. Francini, S. Lepsoy, and M. Balestri, "CDVS: Telecom Italias response to CE1 interest point detection," *ISO/IEC JTC1/SC29/WG11/M30256*, Jul 2013.

[5] Z. Wang, L.-Y. Duan, J. Lin, J. Chen, T. Huang, and W. Gao, "Optimizing binary fisher codes for visual search," in *2015 Data Compression Conference*. IEEE, 2015, pp. 475–475.

[6] "Call for Proposals for Compact Descriptors for Video Analysis (CDVA)-Search and Retrieval," *ISO/IEC JTC1/SC29/WG11/N15339*, 2015.

[7] Y. Lou, F. Gao, Y. Bai, J. Lin, S. Wang, J. Chen, G. Chunsheng, V. Chandrasekhar, L. Duan, T. Huang, and A. Kot, "Improved retrieval and matching with CNN feature for CDVA," *ISO/IEC JTC1/SC29/WG11/m39219*, 2016.

[8] Y. Lou, Y. Bai, J. Lin, S. Wang, J. Chen, V. Chandrasekhar, L. Duan, T. Huang, A. Kot, and W. Gao, "Compact deep invariant descriptors for video retrieval," *Data Compression Conference*, 2017.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, "Fast convolutional nets with fbfft: A gpu performance evaluation," *arXiv preprint arXiv:1412.7580*, 2014.

[11] M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov, "Parallel computing experiences with cuda," *Micro, IEEE*, vol. 28, no. 4, pp. 13–27, 2008.

[12] C. Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)," 2007.

[13] B. Rister, G. Wang, M. Wu, and J. R. Cavallaro, "A fast and efficient sift detector using the mobile gpu," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 2674–2678.

[14] C. Lee, C. E. Rhee, and H.-J. Lee, "Complexity reduction by modified scale-space construction in sift generation optimized for a mobile gpu," *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.

[15] G. Wang, B. Rister, and J. R. Cavallaro, "Workload analysis and efficient opencl-based implementation of sift algorithm on a smartphone," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, 2013, pp. 759–762.

[16] D. Patlolla, S. Voisin, H. Sridharan, and A. Cheriyadat, "Gpu accelerated textons and dense sift features for human settlement detection from high-resolution satellite imagery," *GeoComp*, 2015.

[17] N. Cornelis and L. Van Gool, "Fast scale invariant feature detection and matching on programmable graphics hardware," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*. IEEE, 2008, pp. 1–8.

[18] W. Ma, L. Cao, L. Yu, G. Long, and Y. Li, "Gpu-fv: Realtime fisher vector and its applications in video monitoring," *arXiv preprint arXiv:1604.03498*, 2016.

[19] V. Volkov, "Better performance at lower occupancy," *GTC 2010*, 2010.

[20] M. Lu, "Fast implementation of scale invariant feature transform based on cuda," *Appl. Math*, vol. 7, no. 2L, pp. 717–722, 2013.

[21] "MPEG CDVS (Compact Descriptors for Visual Search) Dataset," http://pacific.tilab.com/www/datasets/download/Dataset-20120210/, accessed Nov. 17, 2015.

[22] "MPEG CDVS (Compact Descriptors for Visual Search) Benchmark," http://purl.stanford.edu/qy869qz5226, accessed Nov. 17, 2015.

[23] M. J. Huiskes, B. Thomee, and M. S. Lew, "New trends and ideas in visual concept detection: The mir flickr retrieval evaluation initiative," *Proc. ACM Int. Conf. Multimedia Inf. Retr.*, pp. 527–536, 2010.

[24] L. Duan, W. Sun, X. Zhang, S. Wang, J. Chen, J. Yin, S. See, T. Huang, A. C. Kot, and W. Gao, "Fast MPEG-CDVS Encoder with GPU-CPU Hybrid Computing," *arXiv preprint arXiv:1705.09776*, 2017.