High-performance Out-of-Core Randomized Singular Value Decomposition on GPU

Yuechao Lu, Fumihiko Ino and Yasuyuki Matsushita
Department of Computer Science, Osaka University, Osaka, Japan
{yc-lu, ino, yasumat}@ist.osaka-u.ac.jp

Summary

Randomized Singular Value Decomposition (SVD)[1] is gaining attention in finding structure in scientific data. However, processing large-scale data is not easy due to the limited capacity of GPU memory. To deal with this issue, we propose RLAGPU, an out-of-core process method accelerating large-scale randomized SVD on GPU.

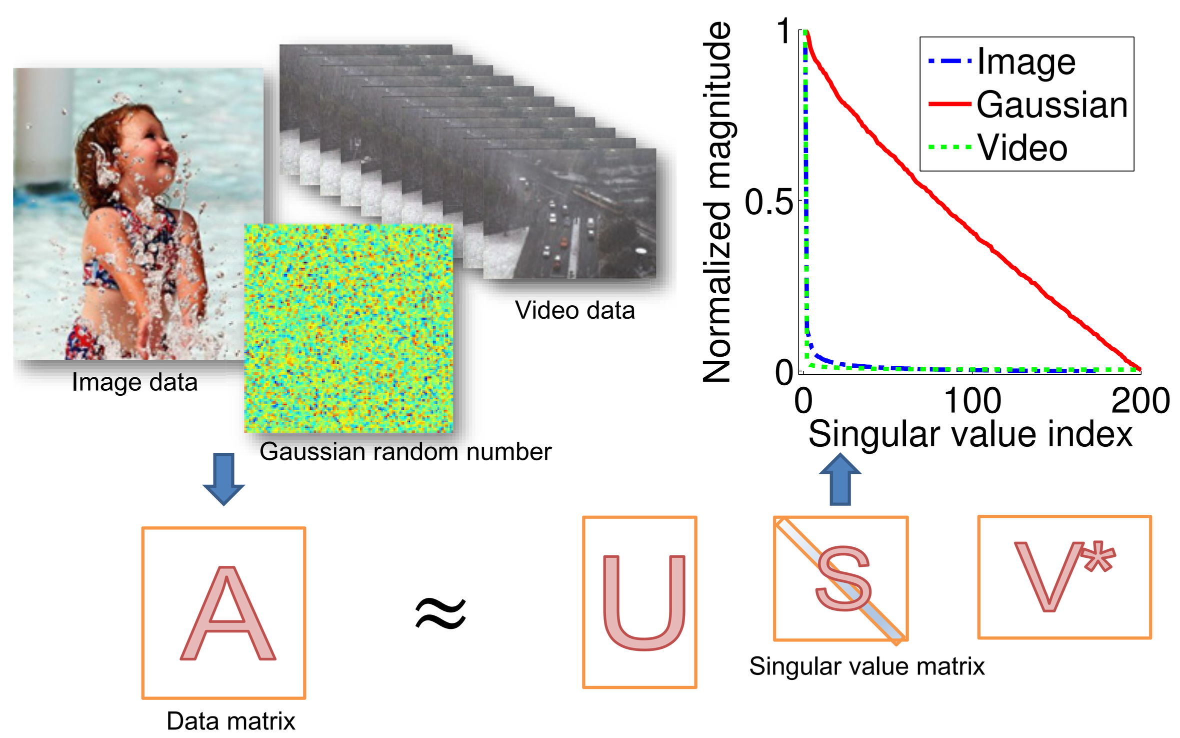
The contribution of our method is as follows:

- Out-of-core implementation that overcomes the GPU memory capacity limit.
- High-performance. In-core and out-of-core routines switched automatically according to data size and available GPU memory.

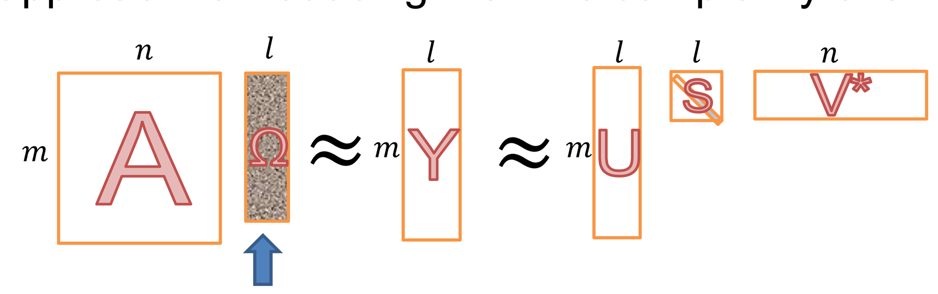
We found that our proposed method outperforms the existing cuBLAS-XT by a margin up to 50%.

Why Out-of-core Randomized SVD?

- Low-rank matrix approximation exists in a lot of problems like data mining, information retrieval, machine learning, bioinformatics, etc. SVD is a common matrix decomposition method for finding singular values in low-rank matrix.



- Decomposing a low-rank matrix by randomized SVD (rSVD)[1] is an emerging approach for reducing the time complexity of a full SVD.



- In order to utilize fast BLAS computation on GPUs, an out-of-core method is necessary for processing large matrices on a limited GPU memory.

Related Work

- Yamazaki, *et al.* [2] proposed randomized SVD on a hybrid CPU/GPU cluster. Their work shows random sampling algorithm obtain speedups of up to 14.1 x in a cluster environment.
- Voronin, *et al* [3] proposed a comprehensive randomized linear algebra library called RSVDPACK. Their GPU implementation is limited by the capacity of GPU memory, which can process data up to 0.5 GB with 12 GB memory.

Reference

[1] Halko, Nathan, Per-Gunnar Martinsson, and Joel A. Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." *SIAM review* 53.2 (2011): 217-288.

[2] Yamazaki, Ichitaro, et al. "Random Sampling to Update Partial Singular Value Decomposition on a Hybrid CPU/GPU Cluster." SC15, November 15-20, 2015.

[3] Voronin, Sergey, and Per-Gunnar Martinsson. "RSVDPACK: Subroutines for computing partial singular value decompositions via randomized sampling on single core, multi core, and GPU architectures." *arXiv preprint arXiv:1502.05366* (2015).

Randomized SVD Algorithm

Algorithm 1: Randomized SVD [1]

Input : matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, target rank k, oversampling parameter p, and power iteration exponent q
Output : SVD of $\mathbf{A}$: matrices $\mathbf{U} \in \mathbb{R}^{m\times l}$, $\mathbf{\Sigma} \in \mathbb{R}^{l\times l}$, and $\mathbf{V}^\top \in \mathbb{R}^{l\times n}$

```
Generate a Gaussian matrix Ω ∈ R^{n×l}, where l = k + p.
Y = (AA^T)^q AΩ ;          // sketch A and perform power iterations
Q = orthonormalize(Y) ;    // form an orthonormal basis of Y
B = Q^T A ;                // form B
[Ũ, Σ, V^T] = svd(B) ;     // truncated rank-l SVD of B
U = QŨ ;                   // form U
```

BRSVD: Block Randomized SVD

- Input data A is partitioned into small blocks.
- Block data reuse for batched power iteration to minimize data transfer between CPU and GPU.
- Automatic block size choosing.

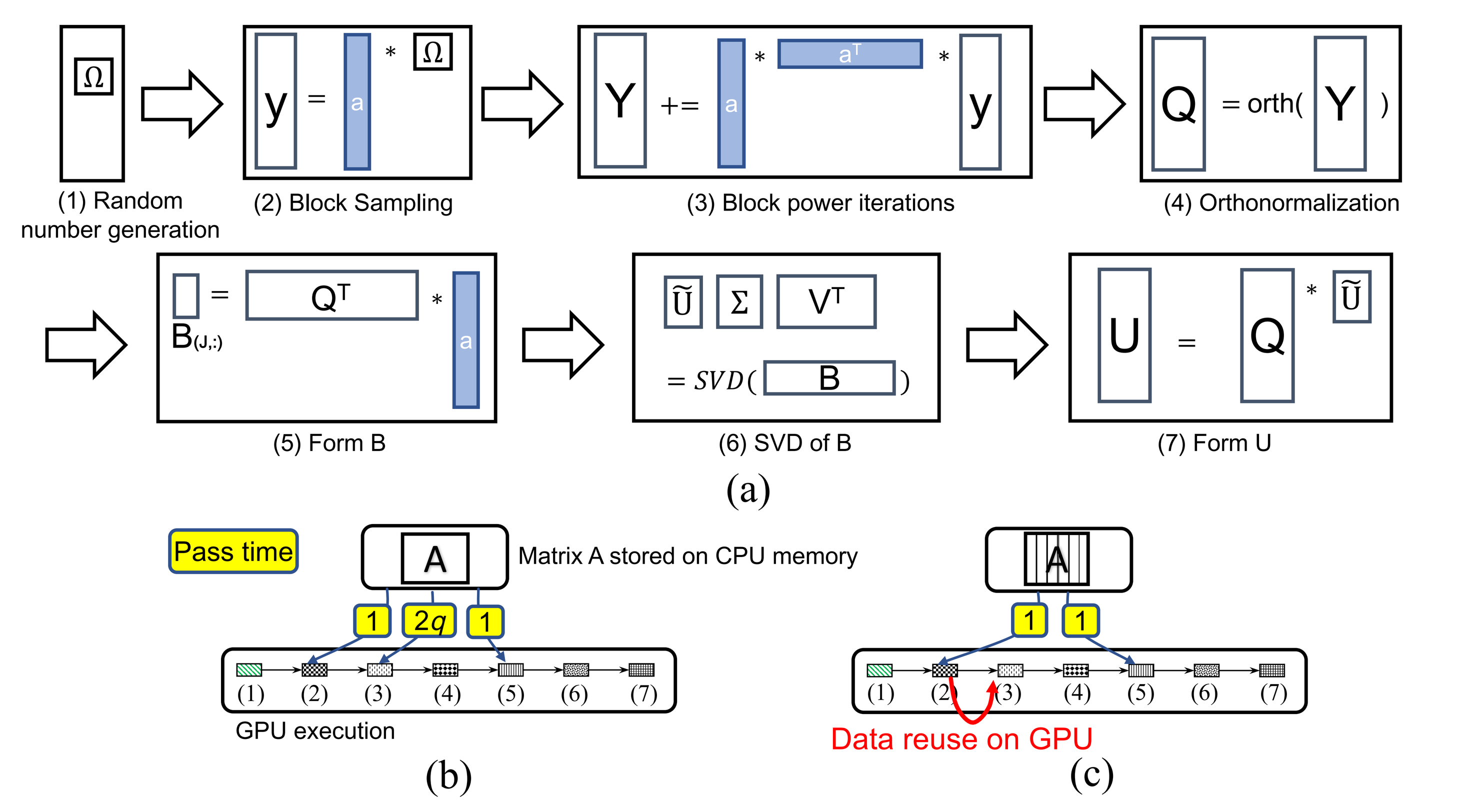


Figure 1: (a) Proposed block randomized SVD (BRSVD) method. A column block **a** is reused on a GPU in the RSVD computation pipeline. *Pass-efficiency* comparison of (b) naïve implementation and (c) proposed BRSVD illustrates that our method reduces data accesses for efficient computation.

Algorithm 2: Proposed method: RSVD by block column sampling

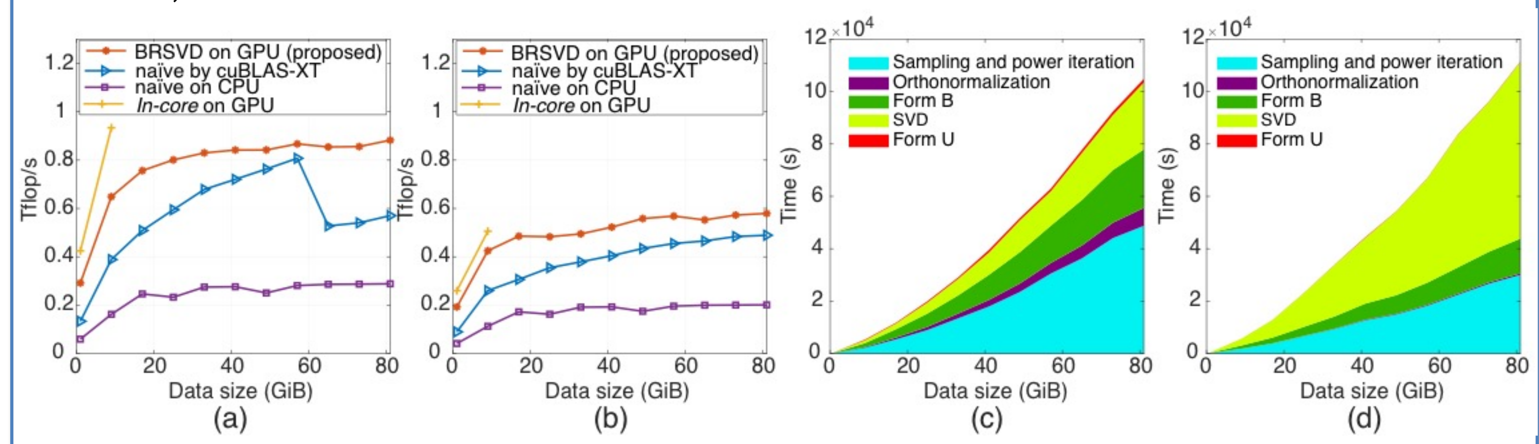
Input : matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, target rank k, oversampling parameter p, power iteration exponent q, and partition number s
Output : SVD of $\mathbf{A}$: matrix $\mathbf{U} \in \mathbb{R}^{m\times l}$, $\mathbf{\Sigma} \in \mathbb{R}^{l\times l}$, and $\mathbf{V}^\top \in \mathbb{R}^{l\times n}$

```
n' = ⌈n/s⌉;  l = k + p;  Y = 0_{m×l};
for j ← 0 to s − 1 do                    // J denotes index set jn' : (j+1)n' − 1
    Generate a Gaussian matrix Ω_j ∈ R^{n'×l};
    Y ← Y + (A_{(:,J)} A_{(:,J)}^T)^q A_{(:,J)} Ω_j ;   // sketch A_{(:,J)} and do power iterations
end
Q_{m×l} = orthonormalize(Y) ;            // orthonormalization by CAQR
free Y;  B = 0_{l×n};
for j ← 0 to s − 1 do
    B_{(:,J)} ← Q^T A_{(:,J)} ;          // form B
end
[Ũ, Σ, V^T] = svd(B) ;                   // truncated rank-l SVD of B
U = QŨ ;                                 // form U
```

Experimental Result

- **Environment**

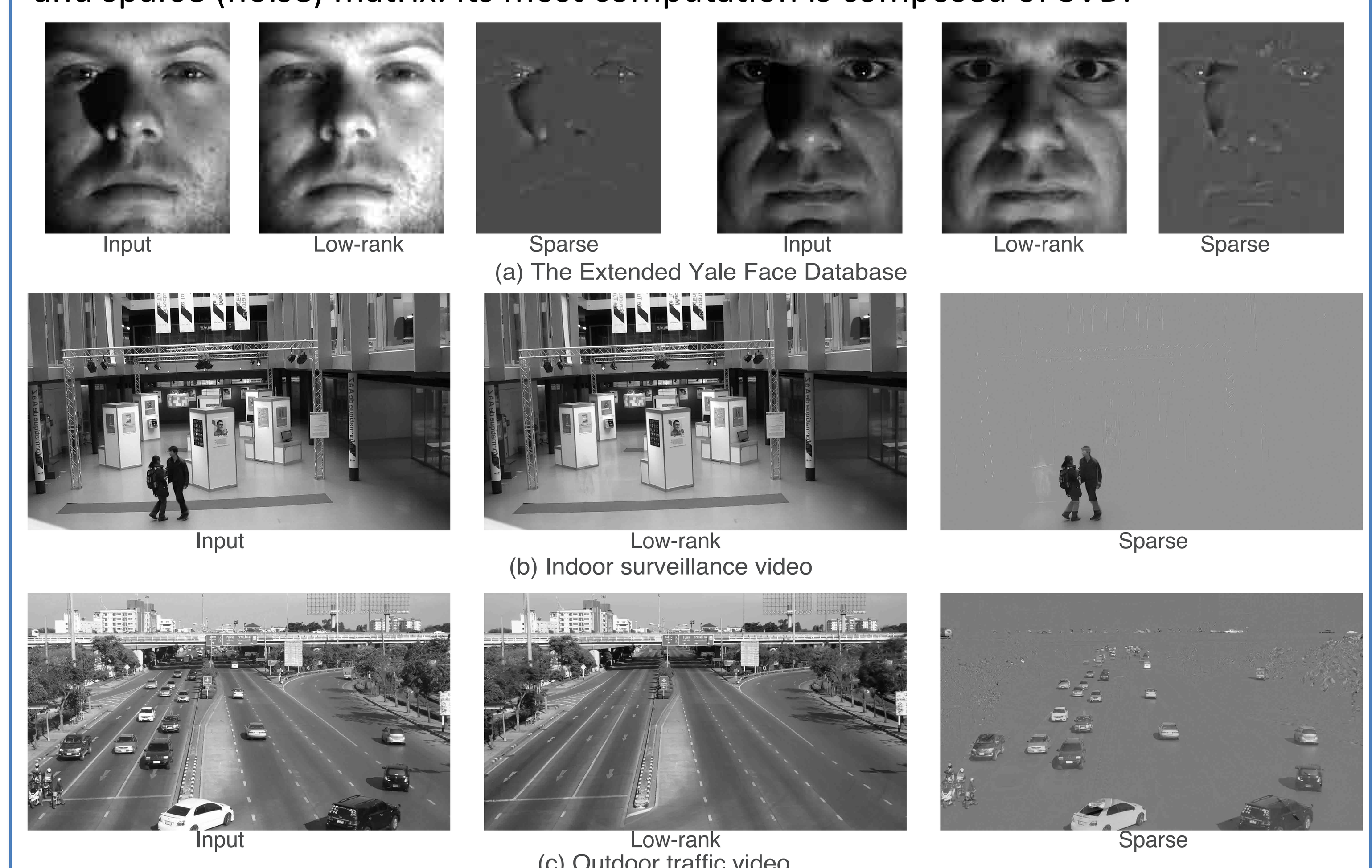
- CPU: Math Kernel Library 11.1.1 on two-socket Intel 10-core Xeon E5-2650v3
- GPU: CUDA 8.0 on Pascal P100 (16 GB memory)
- 4 implementation: CPU, In-core on GPU, naïve (without data partition) by cuBLAS_XT on GPU, BRSVD on GPU.



(a) and (b) show overall performance of tall-skinny matrices (m : n : k = 1024 : 32 : 1) and square (m : n : k = 256 : 256 : 1) ones in Tflops/s. (c) and (d) show the time breakdown of proposed method for tall-skinny and square matrices.

Application to Robust PCA

Robost PCA is a iterative algorithm which separate the input data into a Low-rank and sparse (noise) matrix. Its most computation is composed of SVD.



Data	Image #	Size (GB)	*tol*		Iterations	RSVD time (s)	Total time (s)	Speedup
The Extended Yale Face	2383	0.57	10^{-7}	CPU	30	4.0	26.6	-
(168×192)			10^{-7}	GPU	30	1.1	2.0	13.3×
Indoor surveillance	100	1.5	10^{-7}	CPU	30	50.6	121.4	-
(1920×1080)			10^{-7}	GPU	30	3.4	5.3	22.9×
Outdoor traffic	100	1.5	10^{-5}	CPU	25	51.2	104.7	-
(1920×1080)			10^{-5}	GPU	24	2.9	4.3	24.3×

Future Work

- Randomized SVD on multi-GPU.
- Randomized linear algebra algorithms including QR, CUR, RPCA is under development.
- Open source code is released at https://github.com/luyuechao/.

This study was supported in part by the Japan Society for the Promotion of Science KAKENHI Grant Numbers 15K12008, 15H01687, and 16H02801.