A Randomized Singular Value Decomposition Algorithm for Image Processing Applications

Eleni Drinea[1] Petros Drineas[2] Patrick Huggins[2]

[1] Computer Science Department, Harvard University
Cambridge, MA 02138, USA
[2] Computer Science Department, Yale University
New Haven, CT 06520, USA

Abstract. The main contribution of this paper is to demonstrate that a new randomized SVD algorithm, proposed by Drineas et. al. in [4], is not only of theoretical interest but also a viable and fast alternative to traditional SVD algorithms in applications (e.g. image processing). This algorithm samples a constant number of rows (or columns) of the matrix, scales them appropriately to form a small matrix, say S, and then computes the SVD of S (which is a good approximation to the SVD of the original matrix). We experimentally evaluate the accuracy and speed of this algorithm for image matrices, using various probability distributions to perform the sampling.

1 Introduction

In many applications we are given an $m \times n$ matrix A and we want to compute a few of its left (or right) singular vectors. Such applications include data clustering (see [15]), information retrieval (see [9]), property testing of graphs, image processing etc. Singular vectors are usually computed via the Singular Value Decomposition (SVD) of A (see section 2).

There are many algorithms that either exactly compute the SVD of a matrix in $O(mn^2+m^2n)$ time (an excellent reference is [7]) or approximate it faster (e.g. Lanczos methods, see [12]). In [6] and [4] randomized SVD algorithms were proposed: instead of computing the SVD of the entire matrix, pick a subset of its rows or columns (or both), compute the SVD of this smaller matrix and claim that it is a good approximation to the SVD of the initial matrix. Theoretical error bounds for these Monte Carlo algorithms were presented. In this paper we experimentally evaluate the performance of the algorithm in [4] (which is better suited for practical applications) by demonstrating its performance vs. standard SVD algorithms for image matrices. This randomized SVD algorithm returns approximations to the top k right (or left) singular vectors of the image matrix. Our goal is to compare these approximations to the exact singular vectors (as they are computed by traditional non-iterative SVD algorithms). One way to compare them is by computing rank k approximations to the image matrix using both sets of vectors and then compare the results.

A general family of applications for this algorithm is Principal Component Analysis applications (e.g. eigenfaces, see [10] or Latent Semantic Indexing in information

retrieval, see [9]), where a database (of documents, images etc.) that exists in a high dimensional space is projected to a lower dimensional space using SVD. Then, answering a query (that is searching for an instance in the database that is close to the query) amounts to projecting the query to the same low-dimensional space and then finding the nearest neighbor. The projections need not be exact for two reasons: the values of the elements of the database are usually determined using inexact methods (e.g. the colors of an image) *and* the exact projection to the lower dimensional space is not necessary, since we are only interested in a nearest neighbor search.

The main reason behind picking image matrices in order to demonstrate the accuracy and speed of our algorithm is that beyond evaluating numerical results (i.e. the relative error of the approximation), we can also estimate the accuracy of the approximation visually! Also, our goal is to demonstrate that our methods work well even for relatively small and very dense matrices (up to 1000 rows and columns, density in general close to 1). We will actually see that for these matrices, uniform sampling performs equally well to our more sophisticated sampling methods. The performance of our algorithm has also been examined in [9] using a matrix from information retrieval datasets, but the matrix there was very large (more than 10^5 rows and columns) and less that 10% dense. Finally, singular vectors of image-matrices are quite useful in image processing (e.g. image compression and image restoration, for details see [1],[2],[8] and [11]). With images getting larger and larger, certain applications might not be able to afford the computational time needed for computing the SVD. In the next paragraph we describe such an application in medical imaging.

In dynamic Magnetic Resonance Imaging (MRI) a series of time ordered images is obtained by continually updating image data as changes occur (e.g. monitoring of surgical procedures). In [13, 14] Zientara et. al. investigated the use of SVD for creating and encoding these images. Their technique approximates the top few left or right singular vectors of an initial image and uses them to define "excitation profiles". These profiles are in turn used to create SVD encoded data for the next image in the series. This process apparently is much faster than fully generating the image using state of the art MRI equipment. Recreating the image (that is "decoding" SVD) amounts to a multiplication with the computed singular vectors. The mathematical formulation of this problem is: given an original image-matrix A_0, compute the top k left or right singular vectors (say a matrix U_k). Then, use U_k to generate the SVD encoding of the next image $\tilde{A}_1$. A_1 is now equal to $U_k^T \cdot \tilde{A}_1$. The process repeats itself and generates $\tilde{A}_2, \tilde{A}_3$ etc. One major constraint is the time required by the SVD computation, which can now be reduced using our algorithm.

The paper is organized as follows: in sections 3 and 4 we present the algorithm in a different way than it was presented in [4], more suited for implementation purposes. In section 5 we demonstrate that although the theoretical error bounds are very weak for our image matrices, in practice the algorithm is very efficient and accurate.

2 Background on SVD

Any $m \times n$ matrix A can be expressed as $A = \sum_{t=1}^{r} \sigma_t u^{(t)} v^{(t)^T}$ where r is the rank of A, $\sigma_t, t = 1 \ldots r$ are its singular values (in decreasing order) and $u^{(t)} \in \mathcal{R}^m, v^{(t)} \in \mathcal{R}^n, t =$

$1 \dots r$ are its left and right singular vectors respectively. The $u^{(t)}$'s and the $v^{(t)}$'s are orthonormal sets of vectors. We also remind that $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$ and $|A|_2^2 = \sigma_1^2$.

In matrix notation, SVD is defined as $A = U\Sigma V^T$ where U and V are orthonormal matrices, containing the left and right singular vectors of A, and Σ is a diagonal matrix containing the singular values of A. We note here that $U^T U = I_r$ and $V^T V = I_r$.

If we define $A_k = \sum_{t=1}^{k} \sigma_t u^{(t)} v^{(t)^T}$, then, by the Eckart-Young theorem, A_k is the best rank k approximation to A w.r.t. the 2-norm and the Frobenius norm. Thus, for any matrix D of rank at most k, $|A - A_k|_2^2 \leq |A - D|_2^2$ and $\|A - A_k\|_F^2 \leq \|A - D\|_F^2$. We say that a certain matrix A has a "good" rank k approximation if $A - A_k$ is small w.r.t. to the 2-norm or the Frobenius norm.

From basic Linear Algebra $A_k = U_k \Sigma_k V_k^T = A V_k V_k^T = U_k U_k^T A$, where U_k and V_k are sub-matrices of U, V containing only the top k left or right singular vectors respectively.

3 The randomized SVD algorithm

Given an $m \times n$ matrix A we seek to approximate its right singular vectors. We pick s rows of A, form an $s \times n$ matrix S and compute its right singular vectors. $A^{(i)}$ denotes the i-th row of A as a row vector.

1. **for** $t = 1$ **to** s
   - Pick an integer from $\{1 \dots m\}$, where Prob(pick l) $= p_l$ and $\sum_{l=1}^{m} p_l = 1$.
   - Include $A^{(l)}/\sqrt{sp_l}$ as a row of S.
2. Compute $S \cdot S^T$ and its singular value decomposition. Now, $SS^T = \sum_{t=1}^{s} \lambda_t^2 w^{(t)} w^{(t)^T}$, where λ_t are the singular values of S and $w^{(t)}, t = 1 \dots s$ its left singular vectors.
3. Return $h^{(t)} = S^T w^{(t)} / |S^T w^{(t)}|, t = 1 \dots k$ (the $h^{(t)}$'s are actually the right singular vectors of S). Define also the $n \times k$ matrix H with columns the $h^{(t)}$'s. Now the $h^{(t)}$ are our approximations to the top k right singular vectors of A.

The sampling process that we describe in the above algorithm could be performed either with or without replacement. A theoretical analysis of sampling without replacement is hard, except for the case of *uniform sampling* without replacement.

In the following theorem we describe the quality of $P = AHH^T$ as a rank k approximation to A. We see that the error $\|A - P\|_F^2$ is at most the error of the optimal rank k approximation $A_k = A V_k V_k^T$ (as defined in section 2) plus some additional error that depends on the number of rows that we included in our sample. The proof is combining results and ideas from [6, 4, 5] and is omitted due to space constraints.

Theorem 1. *If P is a rank (at most) k approximation to A, constructed using the above algorithm, then, for any $s \leq m$, with probability at least $1 - \delta$,*

1. *If $p_l = |A^{(l)}|^2 / \|A\|_F^2$ and sampling is performed with replacement,*

$$\|A - P\|_F^2 \leq \|A - A_k\|_F^2 + \sqrt{\frac{4k}{\delta s}} \|A\|_F^2$$

This sampling minimizes the variance for the error of the approximation.

2. *If $p_l = 1/m$ and sampling is performed with replacement,*

$$\|A - P\|_F^2 \leq \|A - A_k\|_F^2 + \sqrt{\frac{4k}{\delta}\frac{m}{s}\sum_{t=1}^{m}|A^{(t)}|^4}$$

3. *If $p_l = 1/m$ and sampling is performed without replacement,*

$$\|A - P\|_F^2 \leq \|A - A_k\|_F^2 + \sqrt{\frac{4k}{\delta}\left(\frac{m}{s} - 1\right)\sum_{t=1}^{m}|A^{(t)}|^4}$$

The first bound is clearly a relative error bound, saying that the relative error of our approximation is at most the relative error of the optimal rank k approximation plus an additional error ($\sqrt{4k/s\delta}$), which is inversely proportional to the number of rows that we include in our sample. The other two bounds imply weaker relative error bounds, depending on how close $\|A\|_F^2$ is to $\sqrt{m\sum_{t=1}^{m}|A^{(t)}|^4}$. For many dense matrices, in practice, these quantities are quite close. This is the case with the image-matrices in our experiments as well.

Comparing these bounds we see that the first one is the tightest. The third one is tighter than the second, since sampling without replacement is performed and a row can not be included in the sample twice (thus we have more information in our sample). Theoretically, the above algorithm seems to require a significant number of rows of A to be picked in order for the error to become small. In practice (see section 5), the algorithm achieves small relative errors by sampling only a fraction of the rows of A. Also, the above theorem implies in an obvious way bounds for $|A - P|_2$.

We could modify the algorithm to pick columns of A instead of rows and compute approximations to the *left* singular vectors. The bounds in Theorem 1 remain essentially the same (rows become columns and m becomes n). P is now equal to RR^TA, where R is an $m \times k$ matrix containing our approximations to the top k *left* singular vectors.

4 Implementation details and running time of the algorithm

An important property of our algorithm is that it can be easily implemented. Its "heart" is an SVD computation of an $s \times s$ matrix. Any fast algorithm computing the top k right singular vectors of such a matrix could be used in order to speed up our algorithm (e.g. Lanczos methods).

The other computationally significant part of our algorithm is the sampling process. Uniform sampling (with or without replacement) takes constant time, since we simply need to generate numbers uniformly at random from 1 to m. Sampling w.r.t. weights though is more interesting. We describe one way of implementing the sampling process, when $p_l = |A^{(l)}|^2/\|A\|_F^2, l = 1 \ldots m$.

First, compute the norm of each row of the matrix A (total cost $O(mn)$ operations). So, we now know $|A^{(l)}|^2$ and $\|A\|_F^2$, $l = 1 \ldots m$. Define $z_i = \sum_{l=1}^{i}|A^{(l)}|^2, i = 1 \ldots m$ and compute s uniformly distributed random numbers $r_j, j = 1 \ldots s$ from $[0, \|A\|_F^2]$.

We search for i such that $z_i < r_j \leq z_{i+1}$ and we include row i in the sample. It is easy to see that Prob(i-th row in the sample)$= |A^{(i)}|^2/\|A\|_F^2$. This step (using binary search) can be performed in $O(s \log m)$ time. We now scale the rows ($O(sn)$ operations) prior to including them in S.

Since s is a constant, the total running time for the preprocessing step amounts to $O(mn)$ operations and the constant hidden in the big-Oh notation is small, i.e. 3. Then, we compute SS^T and its SVD in $O(s^2n + s^3)$ operations. Finally, we need to compute H, which can be done in $O(nsk)$ operations. Since s, k are constants, the total running time of the algorithm is $O(mn+n)$ time. If we assume uniform sampling (with or without replacement) the total running time of the algorithm is $O(n)$. The constant hidden in this big-Oh notation is quite large, since it is dominated by the square of the number of rows that are included in our sample.

5 Experimental results and discussion

5.1 The setup

We implemented our randomized SVD algorithm using 3 different ways of sampling (as described in section 3): sampling w.r.t. the weights of the rows (with replacement), sampling rows uniformly at random (with replacement) and sampling rows uniformly at random (without replacement). We did not use weighted sampling without replacement, since we have no theoretical bounds for it. These experiments returned approximations to the top k right singular vectors. We also implemented the same experiments sampling columns instead of rows. These experiments returned approximations to the top k left singular vectors and we used them to compute rank k approximations to A. Every experiment was run 20 times for each image.

We fixed k for each image s.t. $\|A - A_k\|_F^2/\|A\|_F^2$ is small ($\leq 1.5\%$) and k reasonably small (depending on the size of the image). We varied s (the number of rows or columns that are included in our sample) between k and $k + 80$ (in increments of 10).

We ran our experiments on all images of the MatLab ImDemos directory. We present results for 2 of these images: the well-known baboon image (a 512 by 512 image) and the cameraman image (a 256 by 256 image). We also present results for 2 large and complex images, the Hand with a Sphere image (a 1220 by 835 image) and the Three Spheres image (a 483 by 861 image).

5.2 Measuring the error and the speedup

To measure the error of our approximations to the top k right (or left) singular vectors, we compute the rank k approximation P to A using H (or R, see section 3) and the relative error of the approximation, namely $\|A - P\|_F^2/\|A\|_F^2$. We also compute (for each image) the relative error of the optimal rank k approximation to A, namely $\|A - A_k\|_F^2/\|A\|_F^2$.

The speedup is measured as the ratio of the time spent by the deterministic SVD algorithm to compute the right singular vectors (which are used to compute A_k) over the time needed by our randomized SVD algorithm to compute H or R (which is used to compute P).

5.3 Discussion

We start by briefly explaining the information included in the captions of figures 1-20. The experiments for each image are described using 4 figures. The first figure is a plot of the relative error of our approximation for each sampling method (see section 3) vs. the number of rows (s) that we include in the sample. The speedup for each value of s is also shown. The second figure is the same as the first, but with columns instead of rows (thus approximations to the left singular vectors are computed). All relative error values are out of 100%.The third one shows the optimal rank k approximation to the image and its relative error ($\|A - A_k\|_F^2 / \|A\|_F^2$). The last one shows our randomized rank k approximation to the image, the number of rows (s) included in the sample, its relative error ($\|A - P\|_F^2 / \|A\|_F^2$) and the speedup achieved. We note here that the image is the *average* image over the 20 runs of the experiment (for this particular value of s).

The most important observation is that our methods seem to return very reasonable relative error bounds, much better than the ones guaranteed by Theorem 1, as we can easily see by substituting values for s and k. In all images we can compute low rank approximations with relative error between 2 and 3 times the optimal relative error. It is very interesting that this relative error is returned by low-rank approximations that are computed using uniform sampling without replacement.

This brings us to our second observation: uniform sampling without replacement returns the best results! This happens because for these images $\sqrt{m \sum_{t=1}^{m} |A^{(t)}|^4}$ is very close to $\|A\|_F^2$. Also, uniform sampling with replacement and weighted sampling return almost the same relative error in all cases (the distance between the 2 curves is within the standard deviation of the experiments).

We also observe that for most images row sampling and column sampling return almost the same results. An exception to this observation is the Three Spheres image, whose rows are much longer than its columns. So, sampling by columns returns better speedups but worse error bounds. The tradeoff though is heavily favoring speedup. Thus, in rectangular images (if we have the choice of approximating either the left or the right singular vectors), it seems to be better to sample the "shorter" dimension.

We will not comment on the visual comparisons between the optimal low-rank approximation and our approximation. We leave that to the reader. The speedup is also obvious from the experimental results and it slightly beats our theoretical expectations, mainly because SVD of large matrices takes more time in practice due to memory I/O from the disk. Our algorithm can be seeing as doing *one-pass* through the image-matrix and then performing computations on a small sub-matrix, thus saving memory and time.

6 Open problems

The major open problem that we would like to address (both from a theoretical and an experimental viewpoint) is to compare our algorithms to iterative methods for approximating singular vectors; we could compute the optimal low-rank approximation using e.g. Lanczos methods and then compare it to our low-rank approximation. We

note here that our algorithm can take advantage of Lanczos methods (as described in section 4). Thus, it can be seen as a way of speeding up these techniques!

Also of interest is to use different sampling methods to pick rows for S. An example is to compute the *gradient* of the image and sample rows according to their weights in the *gradient* image. Intuitively, this sampling technique would favor rows with many "color changes", thus capturing more efficiently the details of the image. Even though theoretically such a method would lead to a larger error, in practice we were able to capture many image details that uniform or weighted sampling were unable to grasp.

Acknowledgments : We wish to thank Ravi Kannan for many helpful discussions and the anonymous reviewers for comments that improved the presentation of the paper. The second author was supported by NSF Grant CCR-9820850.

References

1. H.C. Andrews and C.L. Patterson, *Singular Value Decompositions and Digital Image Processing*, IEEE Trans. ASSP, Feb. 1976, pp. 26-53.
2. H.C. Andrews and C.L. Patterson, *Singular Value Decomposition Image Coding*, IEEE Trans. on Communications, April 1976, pp. 425-432.
3. P. Billingsley, *Probability and Measure*, John Wiley & Sons, 1995.
4. P. Drineas, A. Frieze, R. Kannan, S. Vempala and V. Vinay, *Clustering in Large Graphs and Matrices*, ACM-SIAM Symposium of Discrete Algorihtms 1999.
5. P. Drineas and R. Kannan, *Fast Monte-Carlo Algorithms for Approximating Matrix Multiplication*, accepted to the 42nd Symposium on Foundations of Computing, 2001.
6. A. Frieze, R. Kannan and S. Vempala, *Fast Monte-Carlo Algorithms for Low-Rank Approximations*, 39th Symposium on Foundations of Computing, pp. 370-378, 1998.
7. G.H.Golub and C.F.Van Loan, *Matrix Computations*, Johns Hopkins University Press, London, 1989.
8. T. Huang and P. Narendra, *Image restoration by singular value decomposition*, Applied Optics, Vol. 14, No. 9, Sept. 1974, pp. 2213-2216.
9. F. Jiang, R. Kannan, M. Littman and S. Vempala, *Efficient singular value decomposition via improved document sampling*, Technical Report CS-99-5, Duke University, Department of Computer Science, February 1999.
10. B. Moghaddam and A. Pentland, *Probabilistic Visual Learning for Object Representation*, IEEE Trans. Pattern Anal. Mach. Intelligence, 19(7):696:710, 1997.
11. M. Murphy, *Comparison of transform image coding techniques for compression of tactical imagery*, SPIE Vol. 309, 1981, pp. 212 - 219.
12. B. Parlett, *The Symmetric Eigenvalue Problem*, Classics in Applied Mathematics, SIAM, 1997.
13. G. Zientara, L. Panych and F. Jolesz, *Dynamically Adaptive MRI with Encoding by Singular Value Decomposition*, MRM 32:268-274, 1994.
14. G. Zientara, L. Panych and F. Jolesz, *Dynamic Adaptive MRI Using Multi-Resolution SVD Encoding Incorporating Optical Flow-Based Predictions*, Report of National Academy of Sciences Committee on the "Mathematics and Physics of Emerging Dynamic Biomedical Imaging", November 1993.
15. http://cluster.cs.yale.edu/

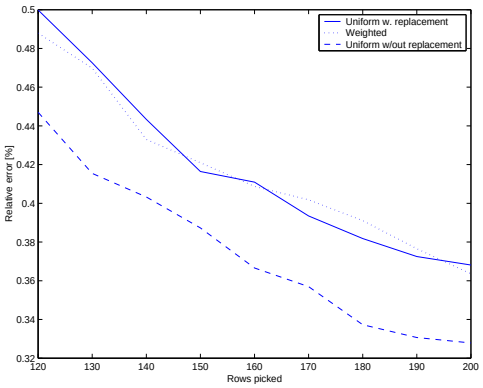

Fig. 1. *Three spheres.* Speedups: 39.9, 33.3, 29.5, 25.4, 22.0, 18.8, 15.9, 13.8, 13.1

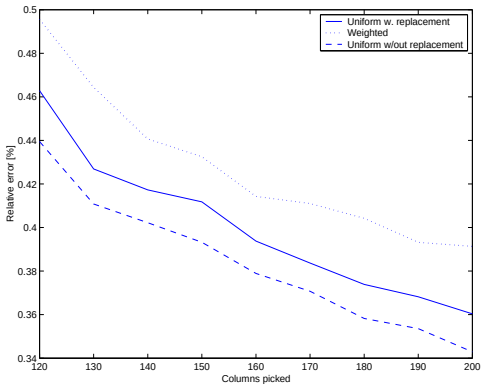

Fig. 2. *Three spheres.* Speedups: 57.0, 48.2, 42.8, 35.4, 30.8, 26.1, 23.7, 19.7, 18.2

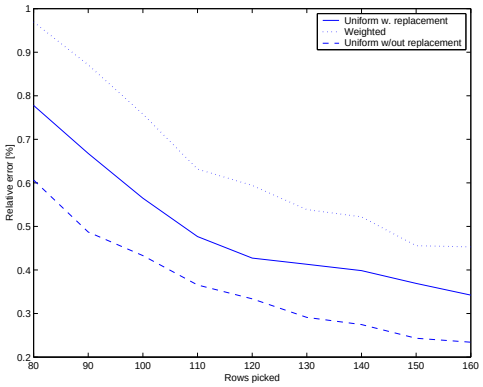

Fig. 3. *Cameraman.* Speedups: 31.0, 28.5, 23.4, 19.1, 16.6, 12.1, 9.8, 8.5, 7.0

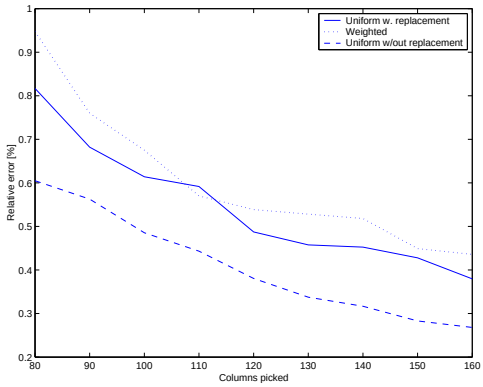

Fig. 4. *Cameraman.* Speedups: 31.7, 29.2, 25.5, 20.4, 17.8, 13.6, 11.6, 9.3, 8.0

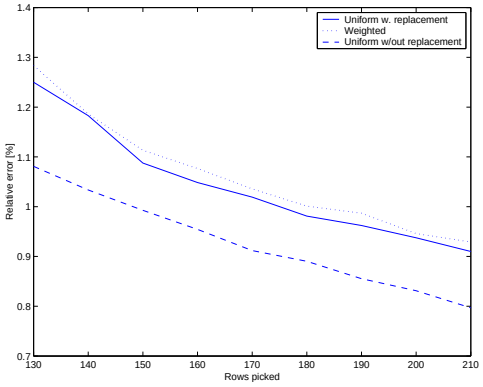

Fig. 5. *Baboon.* Speedups: 64.5, 54.7, 50.6, 43.8, 37.0, 34.6, 29.6, 26.5, 22.7

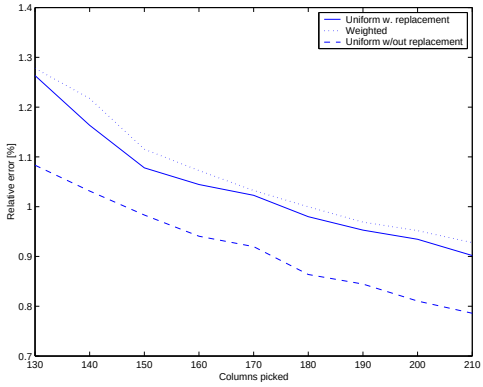

Fig. 6. *Baboon.* Speedups: 77.6, 64.9, 55.9, 47.7, 41.6, 35.0, 30.3, 26.8, 24.4

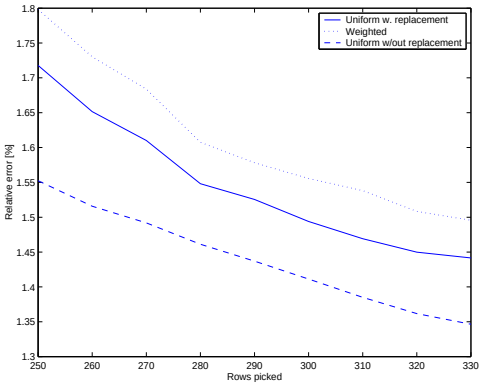

Fig. 7. *Hand with sphere.* Speedups: 28.7, 27.6, 25.2, 23.7, 21.4, 20.2, 18.1, 15.5, 15.5

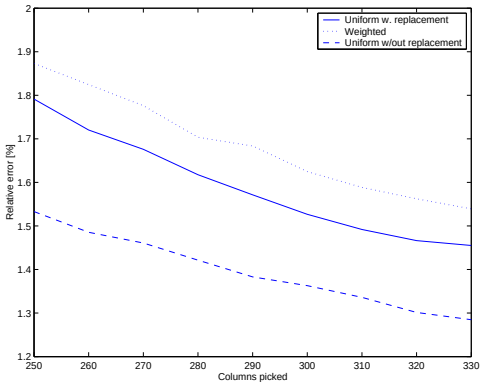

Fig. 8. *Hand with sphere.* Speedups: 26.7, 25.4, 23.0, 21.9, 19.6, 18.2, 16.7, 14.7, 14.4

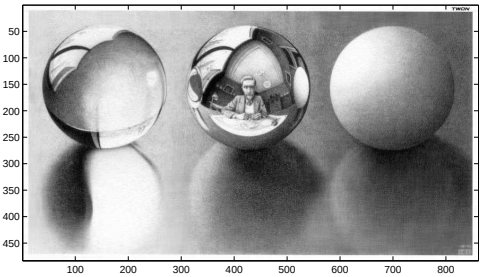

Fig. 9. *Three spheres.* Rank 120 approximation (deterministic), error=0.21%.

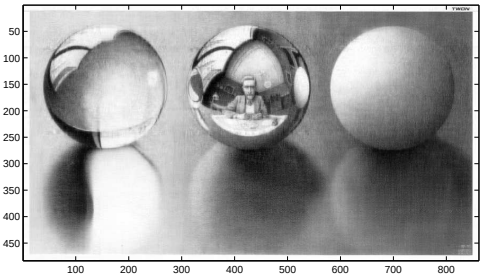

Fig. 10. *Three spheres.* Rank 120 approximation (randomized, using 140 rows), error=0.40%, speedup=29.5

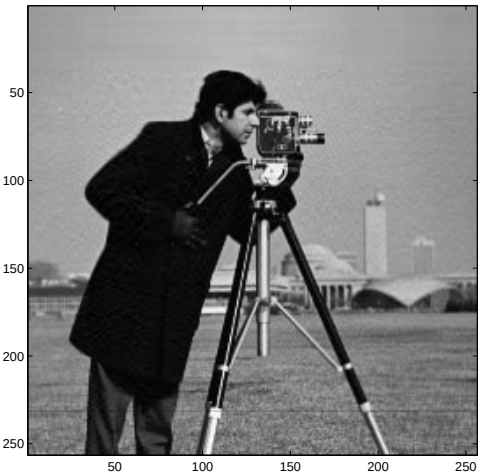

Fig. 11. *Cameraman.* Rank 80 approximation (deterministic), error=0.12%.

Fig. 12. *Cameraman.* Rank 80 approximation (randomized, using 100 rows), error=0.43%, speedup=23.4

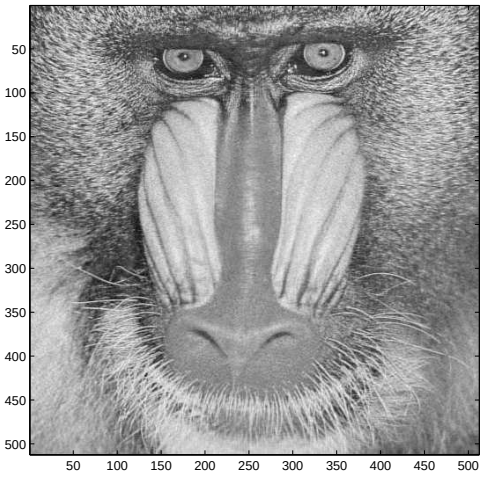

Fig. 13. *Baboon.* Rank 130 approximation (deterministic), error=0.43%.

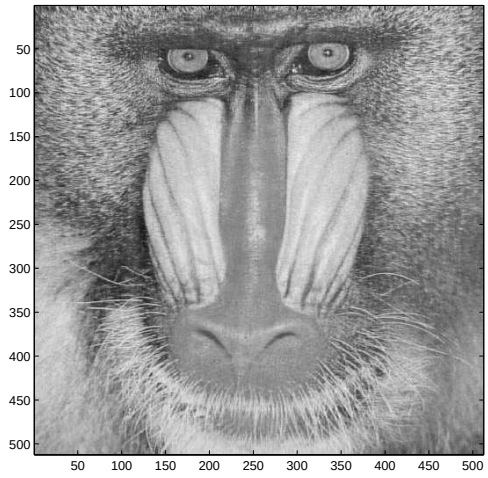

Fig. 14. *Baboon.* Rank 130 approximation (randomized, using 150 rows), error=1.03%, speedup=50.6

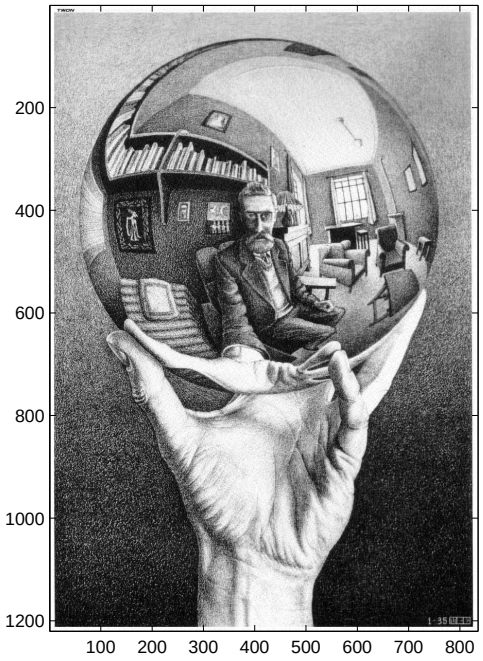

Fig. 15. *Hand with sphere.* Rank 250 approximation (deterministic), error=0.68%.

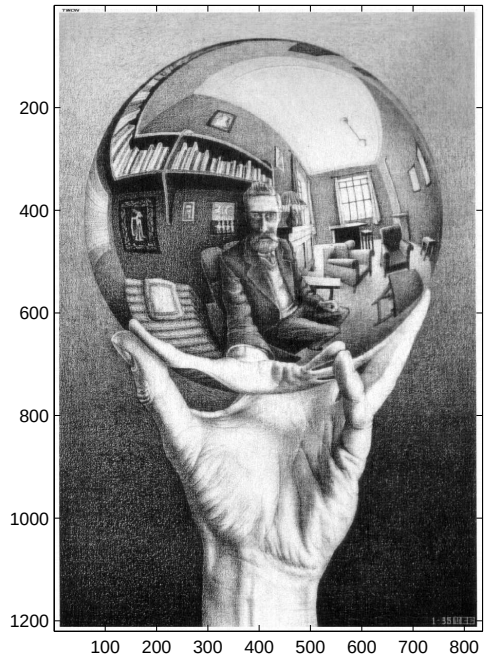

Fig. 16. *Hand with sphere.* Rank 250 approximation (randomized, using 270 rows), error=1.49%, speedup=25.2