Scholarly Paper:
An Alternative Method for Computing the Semidiscrete Matrix Decomposition

Brianne L. Roth
Directed by Dianne P. O'Leary

May 2, 2006

1 Introduction

Digital images can require large amounts of storage to save them within a camera or on a computer. Also, the time to transmit these images, such as over email or when downloading from websites on the internet, can be quite lengthy. By compressing an image, it is able to be stored in a smaller amount of space than the uncompressed version. Obviously, more compressed images can be stored in a fixed amount of space and transmission times are reduced. With the increasing use of digital images, the ability to compress such data continues to be important. As such, methods of image compression are actively being developed and improved.

If we consider an image as a two-dimensional matrix of pixel values (e.g. gray levels), then we can perform a truncated singular value decomposition (SVD) on the matrix to achieve an approximation to the image using less storage than the original. Some other popular compression schemes use a discrete cosine transform (DCT) or discrete wavelet transform (DWT). The SVD chooses basis vectors to fit the problem, while the DCT and DWT constrain the basis vectors to a predetermined set. Similar to the SVD is the semidiscrete decompostion (SDD). The SDD also chooses basis vectors to fit the problem but limits the values that their entries can take.

O'Leary and Peleg [1983] first presented the SDD approximation for application to image compression. Kolda and O'Leary [2000] later expanded it to a weighted SDD approximation and developed software, SDDPACK, that implements their algorithm. They also applied the SDD to latent semantic indexing [1996]. McConnell and Skillicorn [2001] further explored the SDD and gave another application in data mining to find outlier clusters in data sets.

The SDD writes a matrix as a product of three matrix factors, XDY^T. The elements of the matrices X and Y are members of the set $S = \{-1, 0, 1\}$, and D is a diagonal matrix with positive real-valued elements.

A k-term SDD has the form

$$A_k = \underbrace{\begin{bmatrix} x_1 & x_2 & \cdots & x_k \end{bmatrix}}_{X_k} \underbrace{\begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_k \end{bmatrix}}_{D_k} \underbrace{\begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_k^T \end{bmatrix}}_{Y_k^T} = \sum_{i=1}^{k} d_i x_i y_i^T .$$

Each x_i is an m-vector with entries from the set S, each y_i is an n-vector with entries from the set S, and each d_i is a positive scalar.

While an exact decomposition of an $m \times n$ matrix A can be achieved using at most mn terms, we can also construct a k-term approximation. Since this requires storage for just k floating point numbers and $k(m+n)$ entries from S, we achieve excellent compression compared to storage of the dense matrix A if k is small enough.

In Section 2 we review Kolda and O'Leary's algorithm for computing the SDD approximation. Section 3 presents an alternative method using preselected values for the elements of D. We discuss some implementation details in Section 4. Numerical results are given in Section 5.

2 Current Algorithm

Given a matrix A and a matrix of weights W, both of size $m \times n$, the weighted SDD attempts to find an A_k of the form given above that approximates A by minimizing $||A - A_k||_W^2$, where the weighted norm

$$||A||_W^2 \equiv \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 w_{ij}.$$

It will be useful to recall the elementwise product of matrices $(A \circ B)_{ij} \equiv a_{ij} b_{ij}$, as the weighted norm can be written as

$$||A||_W^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} (A \circ A \circ W)_{ij}.$$

2.1 Computing the SDD Approximation

The algorithm for generating the k-term SDD presented by Kolda and O'Leary [2000] computes approximations $A_1, A_2, ..., A_k$ by using an alternating algorithm to find one new (d, x, y) triple at a time. Then, each $A_i = A_{i-1} + dxy^T$.

Begin with $A_0 \equiv 0$ and let $R_k \equiv A - A_{k-1}$ be the residual matrix. Then at each iteration, the (d, x, y) triple must solve

$$\min_{d,x,y} F_k(d, x, y) \text{ subject to } x \in S^m, y \in S^n, d > 0, \qquad (1)$$

where
$$F_k(d, x, y) \equiv \|R_k - dxy^T\|_W^2.$$

It can be shown that this is equivalent to solving

$$\max_{x,y} \tilde{F}_k(x, y) \text{ subject to } x \in S^m, y \in S^n, d > 0, \tag{2}$$

where
$$\tilde{F}_k(x, y) \equiv \frac{\left[x^T (R_k \circ W)\, y\right]^2}{(x \circ x)^T\, W\, (y \circ y)}.$$

Note that d has been eliminated. After finding x and y that satisfy (2), d is set to the optimum value as

$$d^* = \frac{x^T (R_k \circ W)\, y}{(x \circ x)^T\, W\, (y \circ y)}.$$

To find x and y, an alternating algorithm is employed. If a y is chosen and fixed, the problem reduces to

$$\max_x \tilde{F}_k(x, y) = \max_x \frac{\left(x^T s\right)^2}{(x \circ x)^T v}, \tag{3}$$

where $s \equiv (R_k \circ W)\, y$ and $v \equiv W\, (y \circ y)$. Now, (3) is solved for x. Similarly, by fixing x, we can solve for y. The algorithm continues, alternating between solving for x and for y, until convergence. The corresponding d value is computed and the approximation to A is improved to $A_k = A_{k-1} + dxy^T$. Note that this method finds a local rather than a global solution to (1).

3 New Method

McConnell and Skillicorn [2001] showed that the SDD algorithm of Kolda and O'Leary computes values of d that are averages of peaks in the data, and this can lead to blurring of edges in images. The new idea is as follows: Instead of computing the optimal value of d from x and y as above, we preselect a value of d and then compute the corresponding x and y vectors. With a careful choice of d, we can preserve sharp edges in the image.

3.1 Computing SDD Approximation

As in the original algorithm, at each iteration we want to find a triple (d, x, y) to solve
$$\min_{d,x,y} F_k(d, x, y) \text{ subject to } x \in S^m, y \in S^n, d > 0, \tag{4}$$

where
$$F_k(d, x, y) \equiv \|R_k - dxy^T\|_W^2.$$

We first rewrite F_k as

$$F_k(d,x,y) = \|R_k\|_W^2 - 2dx^T(R_k \circ W)y + d^2(x \circ x)^T W\,(y \circ y). \tag{5}$$

Given a choice of d, all values are known except the vectors x and y. We can again use an alternating algorithm to find an approximate solution to the problem. Assuming y is fixed, we write

$$F_k(x,y) = \|R_k\|_W^2 + x^T s + (x \circ x)^T v \tag{6}$$

where $s = -2d(R_k \circ W)y$ and $v = d^2W(y \circ y)$. Since R_k is constant with respect to x, the minimum of F_k occurs at the same x as the minimum of

$$\hat{F}_k = x^T s + (x \circ x)^T v = (x_1 s_1 + |x_1|v_1) + (x_2 s_2 + |x_2|v_2) + ... + (x_m s_m + |x_m|v_m). \tag{7}$$

Now, each x_i occurs in exactly one binomial term. For each, we set x_i to the value that minimizes that term. Hence, we can determine the optimum x for a given y in m steps. As above, we can then alternate between computing x and y until convergence. The full algorithm is given in Figure 1.

4 Implementation Details

We provide Matlab software that is a modification of the SDDPACK developed by Kolda and O'Leary and available at `http://www.cs.umd.edu/users/oleary/SDDPACK/index.html`. Here we discuss decisions made in implementing the new SDD algorithm.

4.1 Choosing Values for D

We first explored using the mode or median of the values in the current R_k as the choices for d at each iteration. We take the median to be the middle element after sorting the absolute values of all nonzero entries in R_k, and we let the mode be the most common value among the nonzeros. We give results using both in the next section.

For the mode, we also explored using a binning technique because regions in images tend to be made up of pixel values that are close together but not necessarily equal. For example, a blue sky may consist of a few values rather than just one, yet we should consider the sky's values as belonging to the same set. To perform this clustering, we use the Matlab `kmeans` function to separate the absolute value of the nonzero entries of R_k into bins. We then choose a representative of the bin with the most entries as the mode. This method of choosing d performs slower in comparison to the other two because the `kmeans` clustering is an iterative algorithm that can be slow to converge. Also, the d values resulting from the clustering were not markedly different from those chosen by using the mode. For these reasons, we do not include results for this method.

```
1. Let $R_k$ denote the residual, and initialize $R_1 \leftarrow A$.
   Let $\rho_k = ||R_k||_W^2$ be the square of the norm of the residual, and initialize
   $\rho_1 \leftarrow ||R_1||_W^2$.
   Let $A_k$ denote the $k$-term approximation, and initialize $A_0 \leftarrow 0$.
   Choose $k_{\max}$, the maximum number of terms in the approximation.
   Choose $\rho_{\min}$, the desired accuracy of the approximation.
   Choose $l_{\max}$, the maximum allowable inner iterations.
   Choose $\alpha_{\min}$, the minimum relative improvement, and set $\alpha > 2\alpha_{min}$.

2. For $k = 1, 2, \ldots, k_{\max}$, while $\rho_k > \rho_{\min}$, do

   (a) Choose $d$ as the mode or median of $R_k$.
   (b) Choose $y$ so that $(R_k \circ W)y \neq 0$..
   (c) For $l = 1, 2, \ldots, l_{\max}$,while $\alpha > \alpha_{\min}$, do

       i. Set $s \leftarrow -2d(R_k \circ W)\, y$, $v \leftarrow d^2 W\, (y \circ y)$.
          Solve $\min_x \; x^T s + (x \circ x)^T v$.

       ii. Set $s \leftarrow -2d(R_k \circ W)^T\, x$, $v \leftarrow d^2 W^T\, (x \circ x)$.
           Solve $\min_y \; y^T s + (y \circ y)^T v$.

       iii. $\beta \leftarrow -2dx^T(R_k \circ W)y + d^2(x \circ x)^T W(y \circ y)$.
       iv. If $l > 1$: $\alpha \leftarrow \dfrac{\beta - \bar{\beta}}{\bar{\beta}}$.
       v. $\bar{\beta} \leftarrow \beta$.

       End $l$-loop.

   (d) $x_k \leftarrow x$, $y_k \leftarrow y$, $d_k \leftarrow d$.
   (e) $A_k \leftarrow A_{k-1} + d_k x_k y_k^T$.
   (f) $R_{k+1} \leftarrow R_k - d_k x_k y_k^T$.
   (g) $\rho_{k+1} \leftarrow \rho_k - \beta$.

   End $k$-loop.
```

Figure 1: Computing a Weighted SDD with chosen d values.

4.2 Starting Vectors

As with Kolda and O'Leary's original alternating algorithm, our revision is highly dependent on the starting y vector. In addition to the four starting y vector choices from SDDPACK, we have included one specific to the choice of d. We initialize y_i to one if the i^{th} column of R_k contains this value of d. Conversely, we set y_i to -1 if the i^{th} column of R_k contains $-d$. All unassigned y_i are zero. We call this initialization SET.

4.3 Convergence

The x and y vectors do not always converge with a particular d and starting y vector. For instances where they do not converge given d as the mode, we restart the algorithm with a new choice of d as the next most common value in the data set. If all values have been exhausted, we perform the old method for one iteration. If the initial d value was chosen as the median, our first restart uses the mode. We then continue as above, choosing the next most common value.

5 Results

We present experiments performed on two images, one with sharp edges of part of a circuit board and one with less well-defined edges of bacteria under a microscope. By design, we expect our algorithm to perform better on the circuit image. The images are given below.

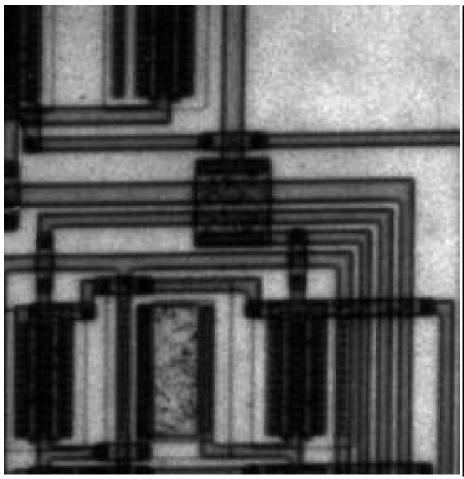


Figure 2: Circuit Image

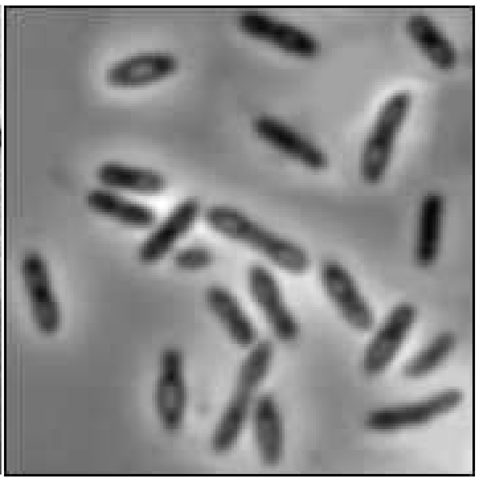


Figure 3: Bacteria Image

We recall the four initialization strategies for the vector y as presented by Kolda and O'Leary:

THR: Cycle through the unit vectors (starting where it left off at the previous iteration) until $\|R_k e_j\|_2^2 \geq \|R_k\|_F^2/n$, and set $y = e_j$. (Threshold)

CYC: Initialize $y = e_i$, where $i = ((k-1) \bmod n) + 1$. (Cycling)

ONE: Initialize y to the all ones vector. (Ones)

PER: Initialize y to a vector such that elements $1, 101, 201, \ldots$ are one and the remaining elements are zero. (Periodic ones)

They showed that THR performs best overall with CYC a close second. Our method was very slow to converge with THR so we have not included it here, but give results for the other initializations. Our new initialization technique, SET, as described above, which sets the y_i's to 1 based on the d value, is also included.

For each image, we perform 25, 50, and 100 term approximations using each of CYC, ONE, PER, and SET to initialize y and each of the two ways of choosing d. We also perform 25, 50, and 100 term approximations with Kolda and O'Leary's method using THR, ONE, and PER to initialize y. (CYC did not converge for the bacteria image so we do not include it for either image here.)

Tables 1 and 2 show the total number of restarts for each using mode and median for choice of d on both images. The algorithm did not ever have to run the old SDD method. We see that using the mode resulted in far fewer restarts than the median. As expected, the circuit image, with its sharp edges, also had somewhat fewer restarts than the bacteria. Overall, CYC and PER also performed fewer restarts than ONE and SET, although ONE and SET did well with d as the mode.

Table 1: Comparison of number of restarts for Bacteria.

Median					Mode				
k-Terms	CYC	ONE	PER	SET	k-Terms	CYC	ONE	PER	SET
25	2	14	0	32	25	2	0	0	0
50	2	64	0	82	50	2	0	0	0
100	2	160	64	180	100	2	0	0	0

Table 2: Comparison of number of restarts for Circuit.

Median					Mode				
k-Terms	CYC	ONE	PER	SET	k-Terms	CYC	ONE	PER	SET
25	0	0	0	42	25	0	0	0	2
50	0	12	0	72	50	0	0	0	2
100	0	108	0	168	100	0	0	0	2

Relative residual norms for tests with both the old and new method are given in Tables 3, 4, and 5. Although the old method performs better in all cases, the residual norms using median are comparable (i.e. within the same order of magnitude). Using mode for the choice of d had a far larger residual in the 25- and 50-term approximations than the other methods for both images. The mode was comparable to the other methods in the 100-term approximation for the circuit image, but was still worse in the 100-term bacteria image by an order of magnitude. Overall, using the median as the d value yields better results than using the mode.

Table 3: Comparison of relative residual norms for mode.

Bacteria					Circuit				
k-Terms	CYC	ONE	PER	SET	k-Terms	CYC	ONE	PER	SET
25	3.952	4.064	2.618	3.988	25	6.273	7.736	5.474	4.507
50	1.976	2.220	1.477	1.996	50	3.274	3.955	3.081	2.351
100	.4669	.7076	.4761	.5856	100	1.137	1.336	1.082	.7649

Table 4: Comparison of relative residual norms for median.

Bacteria					Circuit				
k-Terms	CYC	ONE	PER	SET	k-Terms	CYC	ONE	PER	SET
25	1.038	1.215	1.035	1.471	25	1.593	1.494	1.530	4.836
50	.5514	.5768	.5513	.6147	50	.9509	.7976	.8613	1.132
100	.2977	.1340	.1978	.1238	100	.6303	.3382	.5047	.3385

Table 5: Comparison of relative residual norms for old method.

Bacteria				Circuit			
k-Terms	THR	ONE	PER	k-Terms	THR	ONE	PER
25	.9213	.9316	.9224	25	1.436	1.450	1.422
50	.3576	.3638	.3507	50	.6719	.6783	.6683
100	.1064	.1025	.09912	100	.2848	.2832	.2835

Figures 4 and 5 show the 100-term circuit and bacteria approximations with the smallest residual norms for the old method and the new method using mode and median as compared to the original images. We see that the visual differences between the old method and the new method with d chosen as the median are minimal, while the images with d chosen as the mode are noticeably worse. Both the old and median also give qualitatively accurate approximations to the original.

Figures 6-11 show the progression of 25-, 50-, and 100-term approximations for each method on both images. Our new method performs quite poorly in the 25-term approximation, but improves drastically even with 50 terms.

6 Summary

The performance of our alternative method of computing the SDD approximation is comparable but not better than Kolda and O'Leary's method. Using the median as the choice for d gives a better approximation than using the mode. Exploring other choices for d has the possibility of improving upon this method.

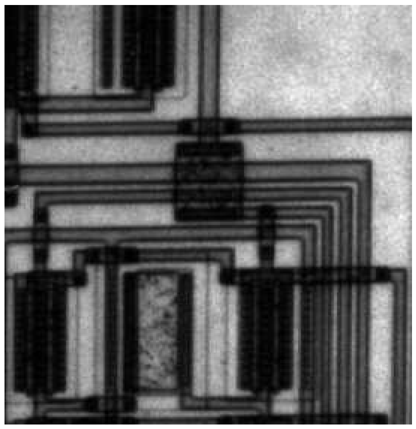
7 References

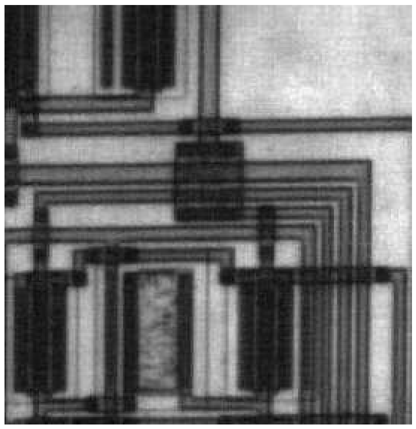
Kolda, T. G. and D. P. O'Leary. 1996. A Semi-Discrete Matrix Decomposition for Latent Semantic Indexing in Information Retrieval. ACM Transactions on Information Systems. 16:322-346.

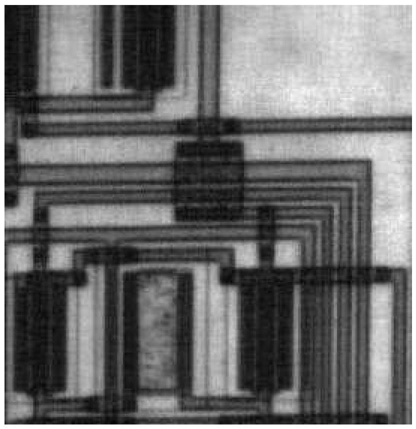
Kolda, T. G. and D. P. O'Leary. 2000. Algorithm 805: Computation and Uses of the Semidiscrete Matrix Decomposition. ACM Transactions on Mathematical Software. 26:415-435.

McConnell, S. and D. B. Skillicorn. 2001. Outlier Detection Using SemiDiscrete Decomposition. External Technical Report. Queens University, Kingston, Ontario, Canada. ISSN-0836-0227-2001-452.

O'Leary, D. P. and S. Peleg. 1983. Digital image compression by outer product expansion. IEEE Transactions on Communications. 31:441-444.


(a) Original circuit image.


(b) Best circuit with old method.


(c) Best circuit with median.

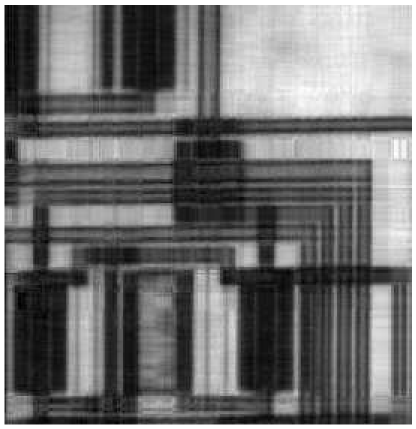
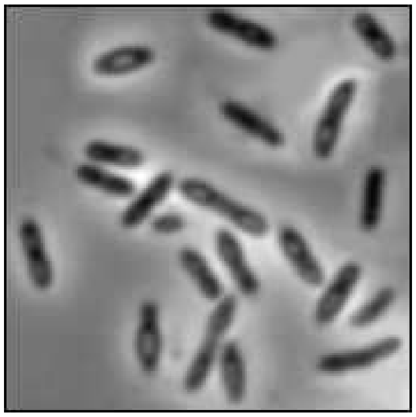
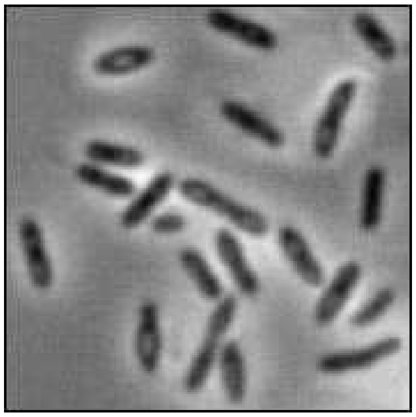
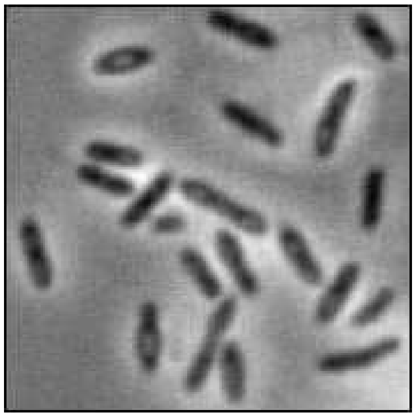

(d) Best circuit with mode.

Figure 4: Comparison of 100-term circuit images with the original.


(a) Original bacteria image.


(b) Best bacteria with old method.


(c) Best bacteria with median.

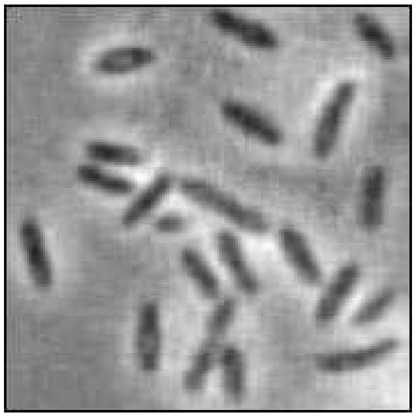
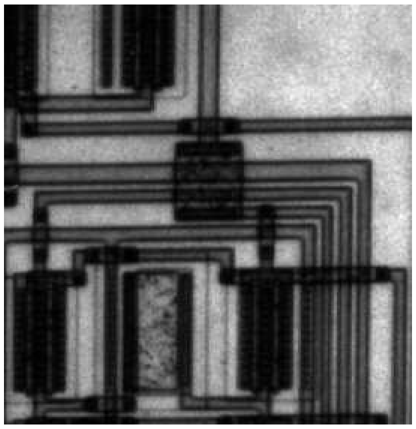
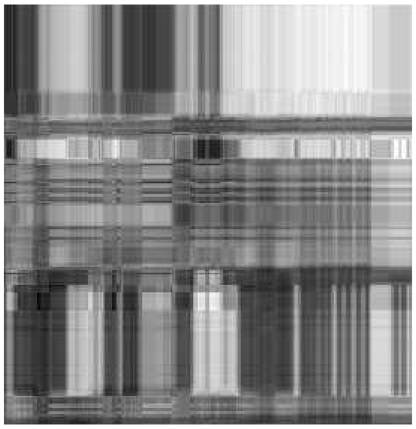

(d) Best bacteria with mode.

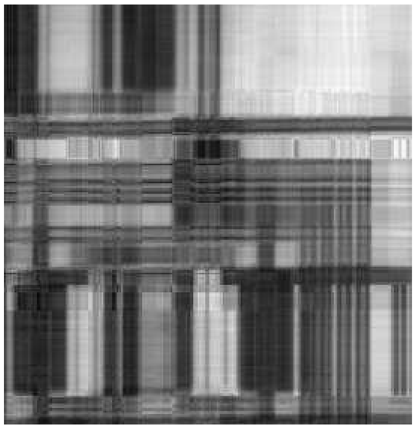
Figure 5: Comparison of 100-term bacteria images with the original.



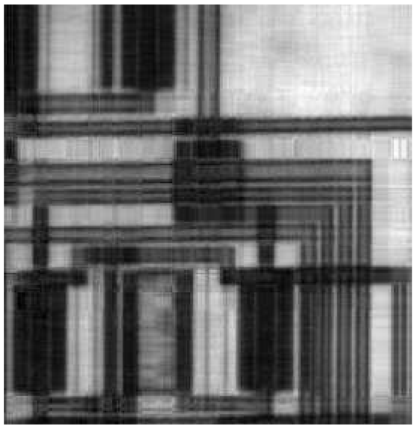
(a) Original circuit image.



(b) 25-term approximation.

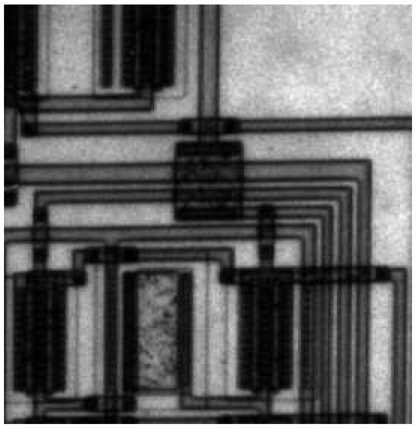


(c) 50-term approximation.

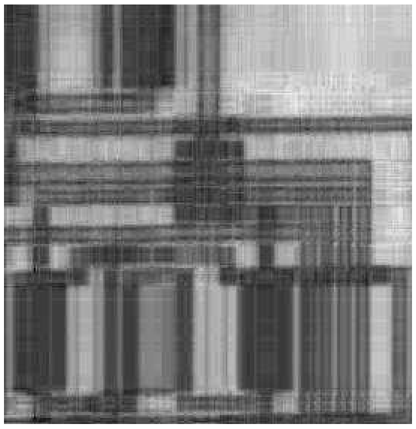


(d) 100-term approximation.

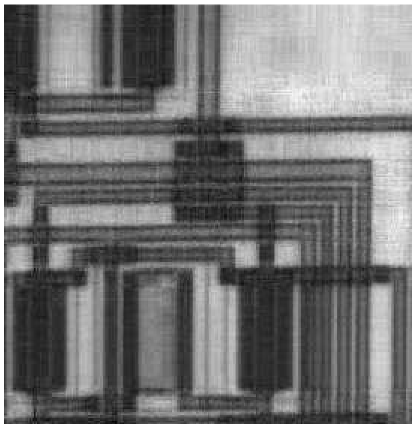
Figure 6: Progression of circuit images using mode to choose d and SET to initialize y.



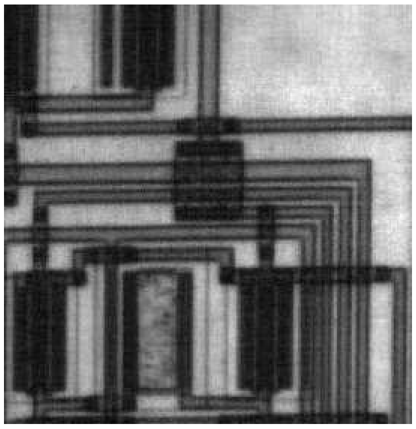
(a) Original circuit image.



(b) 25-term approximation.

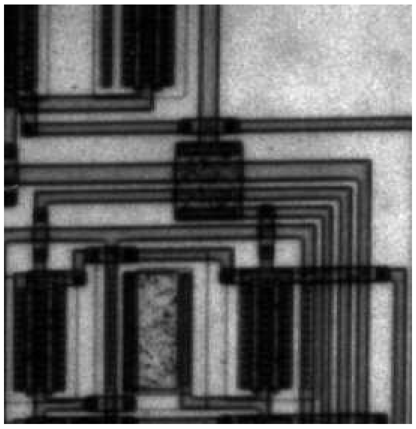


(c) 50-term approximation.

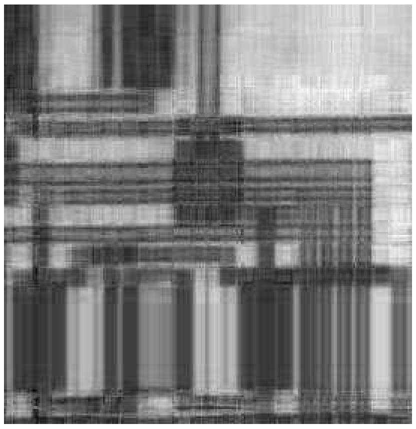


(d) 100-term approximation.

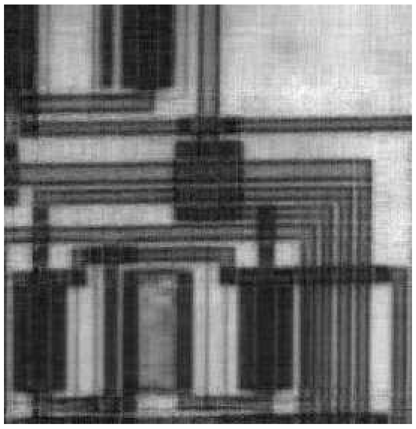
Figure 7: Progression of circuit images using median to choose d and ONE to initialize y.



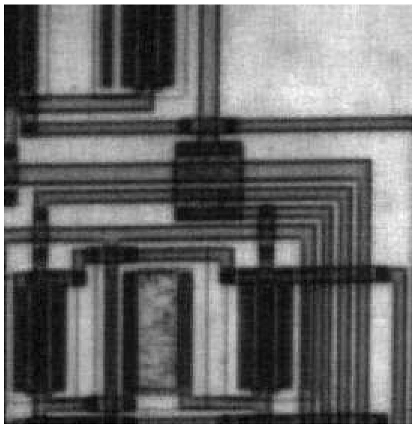
(a) Original circuit image.



(b) 25-term approximation.

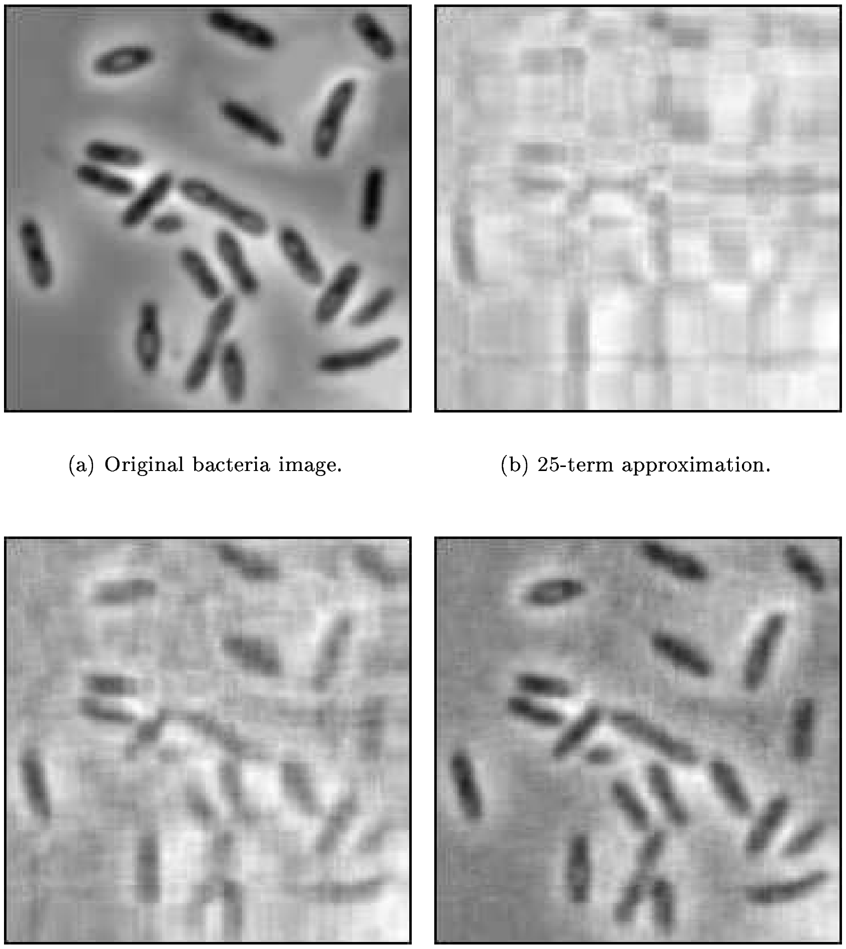


(c) 50-term approximation.



(d) 100-term approximation.

Figure 8: Progression of circuit images using old method and THR to initialize y.



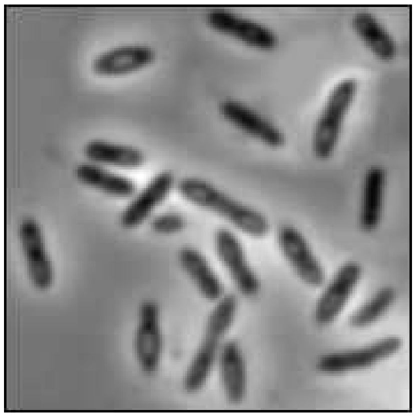
(a) Original bacteria image.

(b) 25-term approximation.

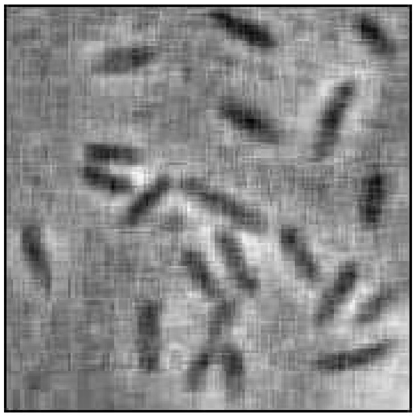
(c) 50-term approximation.

(d) 100-term approximation.

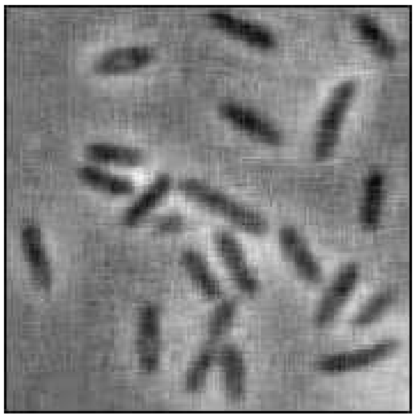
Figure 9: Progression of bacteria images using mode to choose d and CYC to initialize y.



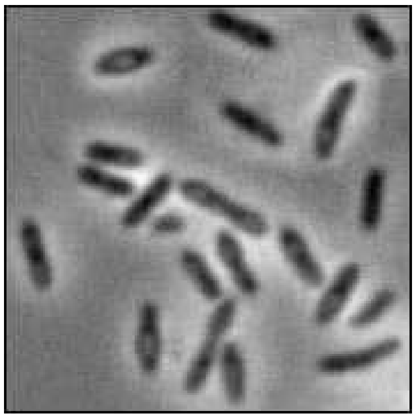
(a) Original bacteria image.



(b) 25-term approximation.

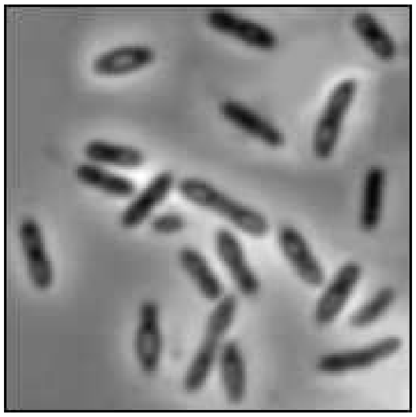


(c) 50-term approximation.

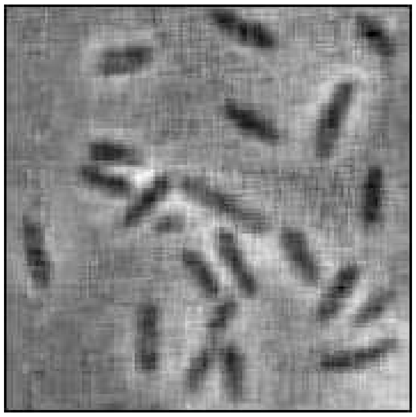


(d) 100-term approximation.

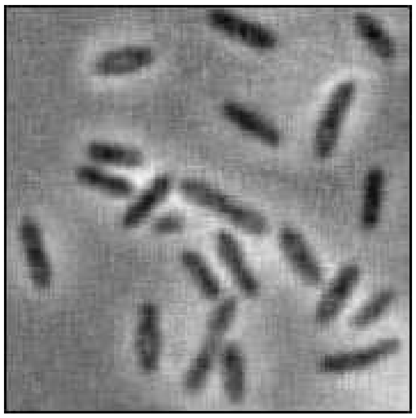
Figure 10: Progression of bacteria images using median to choose d and ONE to initialize y.



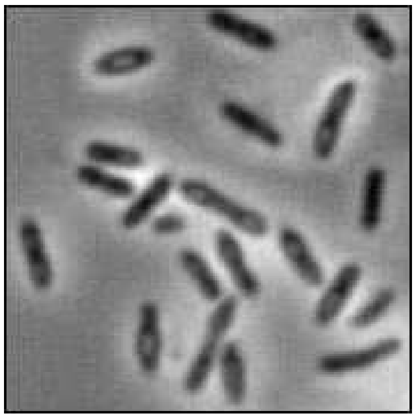
(a) Original bacteria image.



(b) 25-term approximation.



(c) 50-term approximation.



(d) 100-term approximation.

Figure 11: Progression of bacteria images using old method and THR to initialize y.