



**HAL**  
open science

# Generalized Dithering using the Lattice Boltzmann Method

Romain Noël, Alexis Renier-Robin, Laurent Navarro

► **To cite this version:**

Romain Noël, Alexis Renier-Robin, Laurent Navarro. Generalized Dithering using the Lattice Boltzmann Method. *Signal, Image and Video Processing*, 2024, 18 (12), pp.8507-8523. 10.1007/s11760-024-03465-x . hal-04815444

**HAL Id: hal-04815444**

**<https://hal.science/hal-04815444v1>**

Submitted on 2 Dec 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

# Generalized Dithering using the Lattice BOLTZMANN Method

Romain Noël<sup>1\*</sup>, Alexis Renier-Robin<sup>1,2</sup> and Laurent Navarro<sup>2</sup>

<sup>1</sup>Univ. Gustave Eiffel, Inria, Cosys/SII, I4S, Bouguenais, F-44344, France.

<sup>2</sup>Mines Saint-Étienne, Univ. Lyon, Univ. Jean MONNET, INSERM U1059, SAINBIOSE, Centre Ingénierie Santé (CIS), 158 Cours FAURIEL, Saint-Étienne, 42023, France.

\*Corresponding author(s). E-mail(s): [romain.noel@univ-eiffel.fr](mailto:romain.noel@univ-eiffel.fr);

## Abstract

In the present paper, a generalized dithering approach using the Lattice Boltzmann Method (LBM) is proposed. This generalization enables to experiment different models, and to adapt them accordingly to targeted features and problems. As illustrations, 4 models are introduced and tested, including anisotropic gradient-based relaxation time and enhanced equilibrium distribution function, aiming for robust quality and artefacts reduction. These models are implemented on Graphics Processing Units (GPU) and generically interfaced with the OBLiX framework. The several test-cases performed demonstrate that such models yield better qualitative and quantitative results, proving the adaptability of the generalization. This paper, by providing a general framework for dithering using the Lattice Boltzmann Method, allows an adaptation of the algorithms to complex situations deviating from classically encountered cases.

**Keywords:** Dithering, Lattice Boltzmann, Error diffusion, Anisotropic gradient-based

# 1 Introduction

Printing systems are intrinsically capable of only one thing: applying dots of a given color on a physical substrate. For grayscale printing, black dots are used, while color printing utilizes black, cyan, magenta, or yellow dots. To achieve intermediate gray levels or different colors, density or dot size manipulation is necessary. Different density is perceived by the human visual system as gray levels or intermediate colors. For instance, with a grayscale printer, creating a “mid-gray” requires covering 50 % of the area with black dots, which is known as halftoning. Halftoning, fundamental to all printing systems, is often paired with dithering. Dithering involves adding spatial noise to randomize point arrangement, specifically distributing quantization errors from pixel value approximation. Halftoning and dithering exploit the human visual system’s ability to decode information in noise and the retina’s blue noise organization [1]. Blue noise’s power increases by 3 dB per octave with rising spatial frequencies, containing more high frequencies than low frequencies.

The FLOYD-STEINBERG algorithm [2], widely used in dithering, distributes quantization errors to the four nearest right and bottom neighbors. Despite its efficiency with bit-shifting, it suffers from two main drawbacks: worm artifacts and color transition issues. Worm artifacts stem from the symmetry of rightward and downward error diffusion. JARVIS [3] proposed an alternative algorithm utilizing twelve nearest neighbors, reducing artifacts by spreading errors more broadly. STUCKI [4] offers an improved version, similar to [3], with different error distribution weights for enhanced speed. SHIAU and FAN [5] introduced a corrected FLOYD-STEINBERG algorithm with new error diffusion weights, reducing worm artifacts and maintaining bit-shifting capacity. OSTROMOUKHOV [6] proposed further enhancements by using varying coefficients according to gray levels and diffusing errors over three close neighbors, resulting in a more efficient algorithm resembling blue noise distribution. All dithering algorithms are error diffusion-based, distributing density on nearby neighbors. Numerical methods capable of diffusion operations, like LBM,<sup>1</sup> have been utilized in image processing for anisotropic diffusion for years. They are adaptable to dithering.

Another algorithm suggested by JARVIS [3], created shortly after that of FLOYD-STEINBERG, uses twelve nearest neighbors instead of four. The rest of the algorithm is the same as FLOYD-STEINBERG’s one. This algorithm is less efficient, but helps reduce artifacts by spreading the error more widely. An improvement of this algorithm has been proposed in [4]. It is similar as that of [3], but with different error distribution weights which allows a significant improvement in the speed of the algorithm.

The LBM is a method initially developed for gas mechanics, but is now also widely used in fluid mechanics, because it allows the simulation of complex phenomena from particle distribution density. Furthermore, one can

---

<sup>1</sup>Lattice BOLTZMANN Method

show an asymptotic behavior between this method and the NAVIER-STOKES equations [7].

The advantages of the LBM are that it is highly parallelizable, and that it operates on a mesoscopic scale, that is to say on the scale of particle density.

An analogy can be made between particle density and gray levels. Indeed, the gray levels in an image correspond to a concentration of photons that hit some photo-receptors. The LBM can and has been used for image processing for several years. Image processing using LBM is commonly used for denoising [8] and segmentation [9] operations. The anisotropy allowed by the method is one of the advantages that lead to its use.

The LBM can be used for dithering, due to the diffusion process it embeds. In a dithered image, the dots correspond to ink droplets deposited on the paper. As the LBM guarantees the conservation of density, it can also “conserves the ink quantities”. The error diffusion becomes straightforward. In HAGENBURG’s LBM dithering work [10], rotational invariance is preserved, due to the symmetry of the method, and quality in images presenting complex structures is improved. This is, at the cost of a change of scale in terms of calculation times, a significant improvement directly linked to the isotropy of the LBM diffusion algorithm. However, the method is not adapted to colour or multispectral images, plus the relaxation time is unitary, which appears as a limitation. Despite its quality improvement, the use of the LBM for dithering remains quite unexplored, with many questions unanswered. Can generalized anisotropy enhance the quality of LBM based dithering? Can the computational efficiency be improved? In this paper, the main contribution is to perform a generalized LBM dithering that takes into account the anisotropy, inspired by [11]. The algorithm is based on a local variation of the relaxation time or equilibrium function, which becomes dependent on gray levels.

The results show that the artifacts are reduced compared to HAGENBURG. Edges are also improved and better preserved, and grayscale gradients are better captured compared to [12], [2] and [10]. The method is also robust on synthetic and real images. Regarding metrics, GSSIM<sup>2</sup> and MSSIM<sup>3</sup> metrics are found to be better with the proposed method. Finally, the implementation on GPU<sup>4</sup> improves the speed compared to [10].

The paper is organized as follows. Section 2 describes the proposed methodology, based on a generalization of LBM for dithering, modifications of local relaxation times or equilibrium distribution function, and implementation on GPU. The third section is dedicated to the quantitative results using different metrics. The fourth section is a discussion that also presents some qualitative results and observations. The last and fifth section concludes the paper.

---

<sup>2</sup>Global Structural Similarity Index Measure

<sup>3</sup>Mean Structural Similarity Index Measure

<sup>4</sup>Graphics Processing Unit

## 2 Methodology

### 2.1 Generalities about LBM for image processing

As mentioned in the introduction, in the LBM applied to images, the image intensity  $\rho(\mathbf{x}, t)$  is considered and treated as density. Thus, in our context,  $\rho(\mathbf{x}, t)$  is the intensity of an image at time  $t$  and at position  $\mathbf{x} = (x, y)$ . On the other hand, the microscopic state is modelled by a distribution function  $f_i(\mathbf{x}, t)$  which describes the amount of intensity along the direction  $\mathbf{e}_i = \boldsymbol{\xi}_i \Delta t$ . In this paper, we employ a  $D2Q9$  lattice model which means that 9 directions of propagation are considered in 2 dimensions. Using this model, the macroscopic state is obtained as follows:

$$\rho(\mathbf{x}, t) = \sum_{i=0}^8 f_i(\mathbf{x}, t) . \quad (1)$$

The method is based on a collision-propagation scheme at the microscopic scale. The fundamental equation governing the model is

$$f_i(\mathbf{x} + \boldsymbol{\xi}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) , \quad (2)$$

where  $\Omega_i(\mathbf{x}, t)$  is the discrete collision operator. Using the BGK<sup>5</sup> collision operator, the collision-propagation scheme can be written as follows:

$$f_i(\mathbf{x} + \boldsymbol{\xi}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \omega [f_i^{eq}(\mathbf{x}, t) - f_i(\mathbf{x}, t)] , \quad (3)$$

where  $\omega$  is the relaxation time and  $f_i^{eq}(\mathbf{x}, t)$  an equilibrium distribution. Equation (3) can be split into two parts corresponding to two phenomena governing the LBM: collision and propagation on lattice (*i.e.*  $\boldsymbol{\xi}_i \Delta t = \mathbf{e}_i$ ), respectively eq. (4) and eq. (5):

$$f_i(\mathbf{x}, t) = f_i(\mathbf{x}, t) + \omega [f_i^{eq}(\mathbf{x}, t) - f_i(\mathbf{x}, t)] , \quad (4)$$

$$f_i(\mathbf{e}_i, t + \Delta t) = f_i(\mathbf{x}, t) , \quad (5)$$

$$\mathbf{e}_i \in \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\} . \quad (6)$$

### 2.2 A LBM for generalized dithering

The above description explains the general principle of the lattice BOLTZMANN method. By choosing different relaxation time and equilibrium distributions, several treatments on the images can be obtained: denoising, segmentation or dithering [8, 10, 13].

---

<sup>5</sup>BHATNAGAR, GROSS and KROOK

### 2.2.1 Dithering equilibrium function

In 2009, HAGENBURG *et al.* [10] proposed a model to perform dithering on images. Their implementation is based on eq. (3) where the authors have chosen the following parameters:

$$\omega = 1 , \quad (7)$$

and

$$f_i^{eq}(\mathbf{x}, t) = \begin{cases} t'_i \rho(\mathbf{x}, t) & \text{if } \rho(\mathbf{x}, t) < \nu \text{ and } i \neq 0 \\ t'_i (\rho(\mathbf{x}, t) - 255) & \text{if } \rho(\mathbf{x}, t) > 255 \text{ and } i \neq 0 \\ t_i \rho(\mathbf{x}, t) & \text{if } \rho(\mathbf{x} + \mathbf{e}_i, t) > \rho(\mathbf{x}, t) \text{ and } \rho(\mathbf{x} + \mathbf{e}_i, t) < 255 \\ 0 & \text{if } \rho(\mathbf{x} + \mathbf{e}_i, t) < \rho(\mathbf{x}, t) \end{cases} , \quad (8)$$

where

$$t_i = \begin{cases} 4/9 & i = 0 \\ 1/9 & i \in \{1, 2, 3, 4\} \\ 1/36 & i \in \{5, 6, 7, 8\} \end{cases} \quad \text{and} \quad t'_i = \begin{cases} 0 & i = 0 \\ 1/5 & i \in \{1, 2, 3, 4\} \\ 1/20 & i \in \{5, 6, 7, 8\} \end{cases} . \quad (9)$$

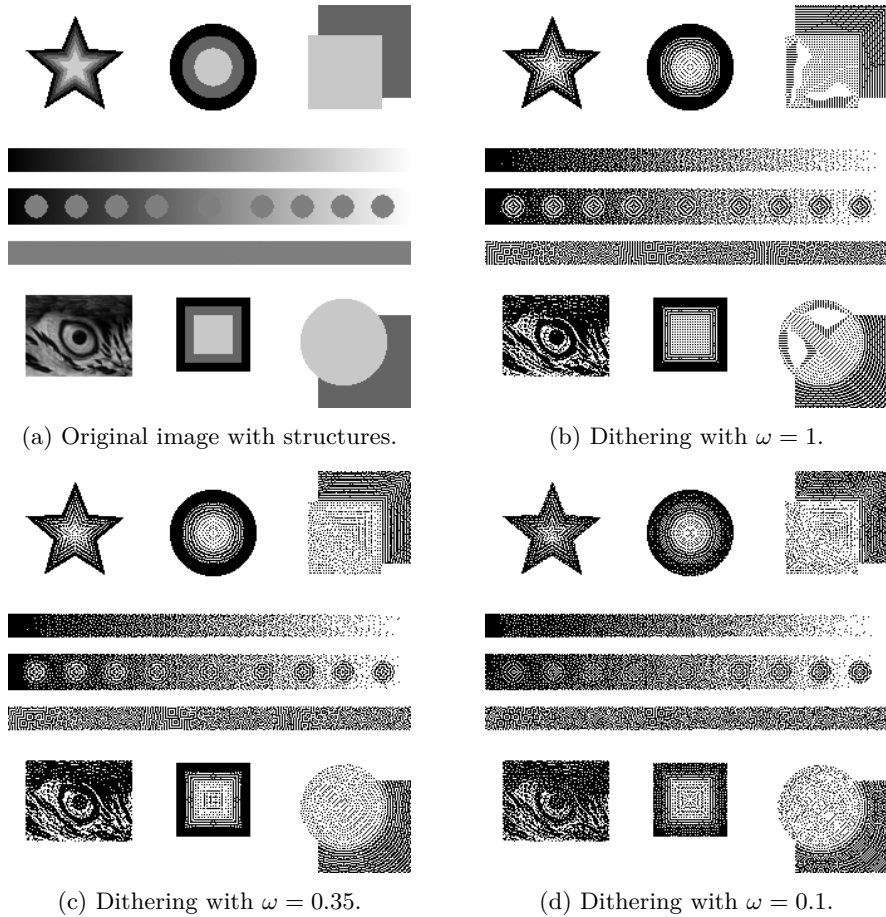
In eq. (8),  $\nu$  is the minimum equilibrium acceptance parameter which is chosen close to zero. This allows a neighboring pixel in  $(\mathbf{x} + \mathbf{e}_i, t)$  to always take away an amount  $t'_i \rho(\mathbf{x}, t)$  of particles of pixel  $(\mathbf{x}, t)$  if the latter has a gray value below  $\nu$ . This parameter is added to improve the robustness of the implementation.

### 2.2.2 Uniform relaxation rate influence

In their paper, HAGENBURG *et al.* have chosen to take the  $\omega$  parameter as unity, making it uniform and isotropic. The result of our implementation of HAGENBURG's algorithm is visible in Fig. 1, we can see that the uniform gray areas suffer from horizontal and vertical bands. As for the gradients, squares are visible in the transition areas. The square shape being probably due to the use of a  $D2Q9$  lattice.

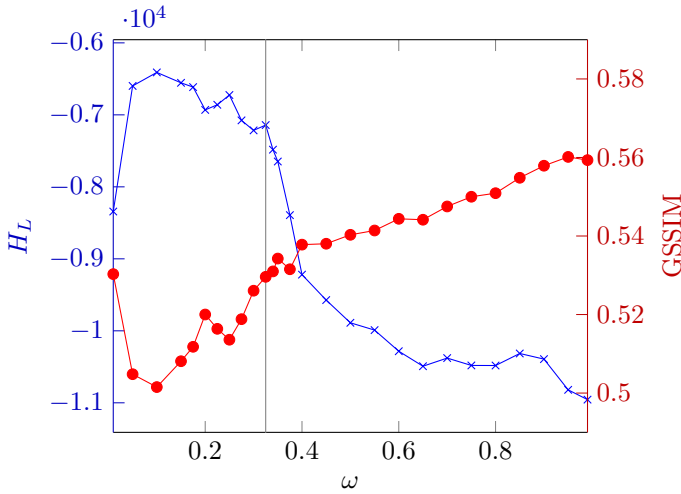
When the relaxation parameter is varied, we observe that it strongly influences the behavior of the method. Letting vary it from 1 to 0.1, for example, produces significant changes, see Fig. 1. It can be seen on Fig. 1(d) that the artifacts in the uniform gray areas have disappeared but at the cost of a blurring of the contours between the areas.

For a uniform relaxation time, we observe that a value of 0.35 seemed to be the switch value between fast and slow diffusion process and thus can be optimal, see Fig. 1(c). This value of 0.35 allows a slow diffusion which gives time for the pixel particles to equilibrate and not to freeze quickly in one state. This avoids artifacts that can be seen in the structures at the top and bottom right of Fig. 1(b). On the other hand, this value is high enough that

6 *Generalized Dithering using the Lattice BOLTZMANN Method***Fig. 1:** Influence of the relaxation parameter.

the diffusion cannot spread too much, thus avoiding the spillover that can be seen on Fig. 1(d). Fig. 2 shows the evolution of two metrics applied only on the gradient part of Fig. 1: the LEOVICI entropy [14] and the GSSIM between the original image and the image dithered with different values of  $\omega$ . In a few words, the first metric gives a measure of the disorder in an image, and it should be as large as possible. Indeed, it is preferred to introduce noise into the dithered image to give the illusion of different gray levels. The latter measures the similarity between two images. It is also better for this metric to be maximal. On Fig. 2, it can be seen that the metrics give diametrically opposed results for image quality as a function of  $\omega$ , which is not inconsistent insofar as the two metrics measure two different phenomena. Moreover, the LEOVICI entropy curve shows that the entropy drops drastically from 0.35

onwards. A lower entropy means less noise in the image and therefore the appearance of the artifacts illustrated above. This confirms the switch value of 0.35 which seems to be an optimal value, allowing us to keep entropy high while maximizing the GSSIM and, therefore, the similarity between the two images.



**Fig. 2:** Evolution of the LEIBOVICI entropy and the GSSIM with  $\omega$ .

Thus, our objective is to build diffusion models, *i.e.*  $\omega$  or equilibrium functions, taking into account the characteristics of the image, to adapt to the particularities of the latter. Models will be both inhomogeneous and anisotropic because of their dependence on both the position in the image and the direction of the transitions in the image.

### 2.3 Testing with various anisotropic relaxation rate functions

In this section, we introduce two relaxation rate functions which have the specificity of being anisotropic because of their dependence on the direction vectors  $e_i$ . More specifically, the distribution is made non-uniform and anisotropic by taking into account the gradient  $\langle e_i | \nabla \rho \rangle$  in the original image.

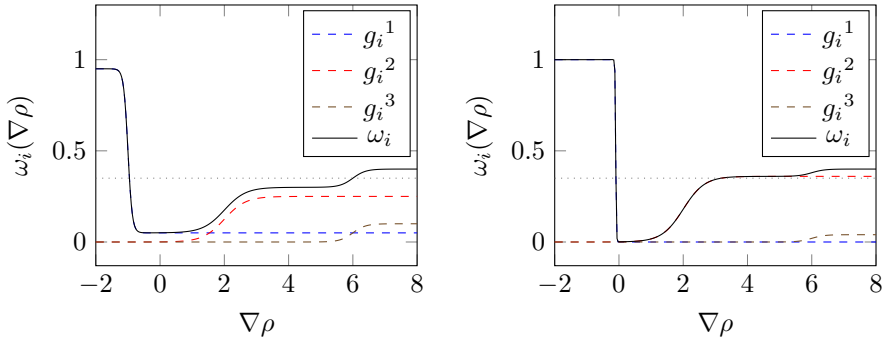
One point to note is that in this case, the lattice BOLTZMANN eq. (3) becomes:

$$f_i(\mathbf{x} + e_i, t + \Delta t) = f_i(\mathbf{x}, t) + \omega_i(\mathbf{x}, t) \cdot [f_i^{eq}(\mathbf{x}, t) - f_i(\mathbf{x}, t)] + \Omega_i(\mathbf{x}, t), \quad (10)$$

where

$$\Omega_i(\mathbf{x}, t) = t_i \sum_{j=0}^8 \omega_j(\mathbf{x}, t) \cdot [f_j^{eq}(\mathbf{x}, t) - f_j(\mathbf{x}, t)]. \quad (11)$$





(a) Model 1.

(b) Model 2.

**Fig. 3:** Proposed models ( $\omega_i(\mathbf{x}, t)$ ) as a function of  $\nabla\rho$ .

The two proposed models for the relaxation rate are composed of sums of 3 functions  $g_i^n$  ( $n \in \{1, 2, 3\}$ ), allowing the handling of different cases of diffusion:

$$g_i^n(\mathbf{x}, t) = a_n + \frac{b_n}{1 + e^{c_n(\langle \mathbf{e}_i | \nabla \rho \rangle + d_n)}}, \quad (12)$$

$$\omega_i(\mathbf{x}, t) = \begin{cases} g_i^1(\mathbf{x}, t) + g_i^2(\mathbf{x}, t) + g_i^3(\mathbf{x}, t) & \text{if } t \leq t_{\text{limit}} \\ 0.5 & \text{otherwise} \end{cases}, \quad (13)$$

where the commonly used value of  $t_{\text{limit}}$  is 20. The form used by  $g_i^n$  in eq. (12), is focusing on the density gradient orientation compared to the lattice directions through a scalar product. While the use of the exponential function tends to nullify positive (and large) values of the scalar product. Therefore, choices for parameters  $a$ ,  $b$ ,  $c$  and  $d$  in eq. (12) are selecting directionally how to diffuse toward the density gradient.

For the following two models, the parameters  $a$ ,  $b$ ,  $c$  and  $d$  of eq. (12) are chosen as constant. As eq. (12) uses only these constants and the gradient of the original image, it implies that eq. (13) is only time-dependent for the change to a constant after  $t_{\text{limit}}$ . This means that the two parts of the relaxation rate can be computed before the LBM iterations start and be switched only once  $t_{\text{limit}}$  has been reached.

### 2.3.1 Model 1: Gradient-based anisotropic diffusion

The model 1, whose representation is visible in Fig. 3(a), has the following parameters:

$$\begin{cases} a_1 = 0.05 \\ b_1 = 0.9 \\ c_1 = 15 \\ d_1 = 1 \end{cases}, \begin{cases} a_2 = 0 \\ b_2 = 0.25 \\ c_2 = -3 \\ d_2 = -2 \end{cases}, \begin{cases} a_3 = 0 \\ b_3 = 0.1 \\ c_3 = -5 \\ d_3 = -6 \end{cases}. \quad (14)$$

The  $g_i^1$  function controls the distribution for negative values of the gradient (*i.e.* a neighboring pixel in the direction  $i$  is darker than the current pixel). Looking at the eq. (8), it can be noticed that the calculation of the equilibrium state prevents the displacement of particles from one pixel to another darker one. This transfer is only allowed by eq. (8) in certain cases, which we then facilitate with a value of 0.9. In order not to stop the diffusion completely,  $g_i^1$  also guarantees a minimum for the distribution even for low gradients.

The function  $g_i^2$  is intended to allow better diffusion as the gradient increases. The sum of  $g_i^1$  and  $g_i^2$  allows to obtain a 0.30 level for a slow diffusion in areas with medium gradients.

For its part,  $g_i^3$  increases the relaxation time for strong gradients to 0.4 for rapid diffusion to take place. This rapid diffusion allows a rapid convergence to a stable state, thus preserving the shape boundaries in the original image and avoiding the spillover effect seen in Fig. 1(d).

Finally, the value of relaxation time becomes uniform with a value of 0.5 after the  $t_{\text{limit}}$  number of iterations. This change is made so that the algorithm can converge to a steady state in all areas, in a reasonable time.

### 2.3.2 Model 2: two-modes anisotropic diffusion

The first model, while performing well, may have difficulty reproducing uniform portions of images, as illustrated in Fig. A6(a). Then, a second model is introduced, with the capacity of handling such cases while keeping performances equivalent to the first model. This model works with two modes, a first classical diffusion mode close to model 1 and a second diffusion mode which forces diffusion for uniform image areas.

As for model 1, model 2, whose representation is visible in Fig. 3(b), has the following parameters:

$$\begin{cases} a_1 = 0 \\ b_1 = 1 \\ c_1 = 100 \\ d_1 = 0.1 \end{cases}, \begin{cases} a_2 = 0 \\ b_2 = 0.36 \\ c_2 = -3 \\ d_2 = -2 \end{cases}, \begin{cases} a_3 = 0 \\ b_3 = 0.04 \\ c_3 = -5 \\ d_3 = -6 \end{cases}. \quad (15)$$

The model variations in  $\omega_i$  are motivated by the same reasons as the first model. However, it introduces a difference in the offset of  $\omega_i(\mathbf{x}, t)$  which is set to zero. This zero offset is to be seen in relation to the additional step

introduced by the model, which is the following:

$$\text{if } \sum_{i=0}^8 \omega_i(\mathbf{x}, t) \leq 1, \text{ then}$$

$$\omega_i(\mathbf{x}, t) = \begin{cases} 0.5 & \text{for } i = 1, 2, 3, 4 \\ \omega_i(\mathbf{x}, t) & \text{for } i = 5, 6, 7, 8 \end{cases} \text{ if } \psi \leq 0.5 \quad (16)$$

$$\omega_i(\mathbf{x}, t) = \begin{cases} \omega_i(\mathbf{x}, t) & \text{for } i = 1, 2, 3, 4 \\ 0.5 & \text{for } i = 5, 6, 7, 8 \end{cases} \text{ if } \psi > 0.5$$

where:

$$\psi \sim \mathcal{U}(0, 1) \quad (17)$$

With this additional step, diffusion is forced for pixels that do not have a preferred diffusion direction (*i.e.* located in a uniform area). This forced random diffusion in directions will “break up” any potential artifacts caused by the lack of diffusion.

## 2.4 Model 3: enhanced equilibrium distribution

The effective management of regions with low gradients is crucial for achieving high-quality dithering, as demonstrated by the two preceding models. These areas are found both in the contours of shapes and in uniform areas of the image, where poor management of low gradients can result in an image with unclear contours and artifacts.

Therefore, in this section, a new equilibrium distribution is introduced, based on eq. (8) proposed by HAGENBURG *et al.* The proposed equilibrium distribution is:

$$f_i^{eq}(\mathbf{x}, t) = \begin{cases} t'_i \rho(\mathbf{x}, t) & \text{if } \rho(\mathbf{x}, t) < \nu \text{ and } i \neq 0 \\ t'_i (\rho(\mathbf{x}, t) - 255) & \text{if } \rho(\mathbf{x}, t) > 255 \text{ and } i \neq 0 \\ t_i \rho(\mathbf{x}, t) & \text{if } \rho(\mathbf{x}, t) \leq \rho(\mathbf{x} + \mathbf{e}_i, t) < \rho(\mathbf{x}, t) + \delta \\ & \text{and } \alpha < \rho(\mathbf{x} + \mathbf{e}_i, t) < \beta \text{ and } (x + y) \equiv 0 \pmod{2} \\ t_i \rho(\mathbf{x}, t) & \text{if } \rho(\mathbf{x}, t) + \delta < \rho(\mathbf{x} + \mathbf{e}_i, t) \text{ and } \rho(\mathbf{x} + \mathbf{e}_i, t) < 255 \\ 0 & \text{otherwise} \end{cases}, \quad (18)$$

where

$$t_i = \begin{cases} 4/9 & i = 0 \\ 1/9 & i \in \{1, 2, 3, 4\} \\ 1/36 & i \in \{5, 6, 7, 8\} \end{cases} \text{ and } t'_i = \begin{cases} 0 & i = 0 \\ 1/5 & i \in \{1, 2, 3, 4\} \\ 1/20 & i \in \{5, 6, 7, 8\} \end{cases}, \quad (19)$$

As eq. (8), the proposed equilibrium distribution forces diffusion for pixels with a value smaller than  $\nu$  and those with a value greater than 255. Similarly, the weights remain unchanged and no diffusion is performed if the value of the neighboring pixel in  $(\mathbf{x} + \mathbf{e}_i, t)$  is not greater than the value of the pixel in  $(\mathbf{x}, t)$ .

The novelty introduced lies in the pixel diffusion in the case where the neighboring pixel value is higher than the pixel value, *i.e.*  $\rho(\mathbf{x} + \mathbf{e}_i, t) \geq \rho(\mathbf{x}, t)$ . On eq. (8), diffusion takes place as soon as the value is higher whereas on the proposed equilibrium, the choice of creating a transition zone is made to improve the diffusion for small differences.

Therefore, an offset parameter  $\delta$  is introduced (typically 5, but can be adjusted according to images). For pixels with small differences from their neighbors (*i.e.*  $\rho(\mathbf{x}, t) \leq \rho(\mathbf{x} + \mathbf{e}_i, t) < \rho(\mathbf{x}, t) + \delta$ ), diffusion only occurs if the sum of coordinates is even. This condition will prevent the appearance of artifacts by creating a “grid” where all the pixels in the image alternate their behavior. Moreover, to prevent the creation of grids in dark or light areas, thus creating artificial pixels, the condition that the pixel value must be between two thresholds  $\alpha$  and  $\beta$  (typically 25 and 230) to receive particles has been added. Finally, if the value is greater than  $\rho(\mathbf{x}, t) + \delta$ , the diffusion takes place as in eq. (8).

## 2.5 GPU implementation

As explained in the introduction, one of the strengths of the LBM method lies in its easy parallelization, which can notably be done on GPUs. Numerous works [15–18] have been carried out to develop optimized solvers for LBM on GPUs, but with a different target than the one of the present work. Indeed, these research projects, like [19, 20], focus on the application of LBM to fluids and not to images. Furthermore, some of them have different features compared to our work, with for example [15] which is developed in python or [18] whose techniques are optimized for 3D geometries. All these reasons justified implementing a GPU parallelization using the OPENACC<sup>6</sup> framework.

### 2.5.1 LBM parallelization

The four stages defining the LBM, namely propagation, collision, calculation of the equilibrium distribution and calculation of the macroscopic state are inherently parallelizable.

The collision and propagation phenomena (eq. (4) and eq. (5)) are fully parallelizable. The calculation in one direction  $\mathbf{e}_i$  of the former depends only on the current state of the distributions. For the latter, the computation of the distribution in direction  $\mathbf{e}_i$  at  $\mathbf{x} + \mathbf{e}_i$  and  $t + \Delta t$  only relies on the previous state of the distribution.

Regarding the calculation of the equilibrium distribution with eq. (8) or eq. (18), although having many conditions, it can be noted that it can also be calculated in parallel. Indeed, the equilibrium distribution in direction  $\mathbf{e}_i$  depends only on  $\rho(\mathbf{x}, t)$  and  $\rho(\mathbf{x} + \mathbf{e}_i, t)$ .

Finally, as each pixel of the macroscopic state in eq. (1) only depends on the distribution at position  $\mathbf{x}$  and  $t$ , the value of each pixel can also be calculated

---

<sup>6</sup>Open ACCelerators

in parallel. It is nevertheless necessary to implement a parallel reduction as the value  $\rho(\mathbf{x}, t)$  is modified in parallel.

### 2.5.2 Parallelization strategy

The implementation of parallelization is done using the model proposed by OPENACC. OPENACC is a directives-based parallel programming model designed for performance and portability. It allows specifying parallelizable areas using directives while letting the compiler determine how to implement them for the desired machine.

Although the GCC<sup>7</sup> compiler supports OPENACC directives, it is still limited. Therefore, the HPC<sup>8</sup> SDK<sup>9</sup> provided by NVIDIA was used. This SDK provides `nvc++`, a NVIDIA C++ compiler, which offers the latest OPENACC features and full support for NVIDIA graphics cards.

In other words, by adding simple directives to existing code, `nvc++` will automatically compile a parallel and optimized code using various techniques such as GPU offloading, interprocedural optimizations and various miscellaneous optimizations. Once the code has been generated, it can be then used on any image thanks to OBLIX.

### 2.5.3 Memory management

In terms of memory, the algorithm relies on four `c++` objects: one object representing a macroscopic state and three objects representing microscopic states (the equilibrium state, the prediffusion state and the postdiffusion state). All these objects use `c++ double` which can hold floating-point values of up to 15 digits taking up a space of 8 bytes in the memory. The use of three microscopic states avoids race condition problems, at the cost of using more memory.

When creating these macroscopic and microscopic objects, the following OPENACC directives are used:

```
1 #pragma acc enter data copyin(this)
2 #pragma acc enter data create(array[0:size])
```

Listing 1: Allocation of device memory

These unstructured data directives indicate to the compiler how much space to allocate on the device to store each object. Once the space has been reserved, the image can be loaded from the host onto the device using the directive:

```
1 #pragma acc update device(array[0:size])
```

Listing 2: Load of the image to the device

The LBM algorithm is performed only on the states present in the device. This avoids data going back and forth between host and device memory, which would slow down the algorithm as moving data at each loop is inefficient.

---

<sup>7</sup>GNU Compiler Collection

<sup>8</sup>High Performance Computing

<sup>9</sup>Software Development Toolkit

One point to emphasize is the calculation of the relaxation rate, which as explained earlier, can be calculated before the LBM algorithm starts. As the computation time of the relaxation rate is negligible compared to the LBM algorithm, it is calculated on the CPU and sent to the GPU. In the case of model 2, it also avoids the need to generate random numbers on the GPU.

### 2.5.4 Loop parallelization

Using `nvc++` and `OPENACC`, the for-loops used to implement eq. (4), eq. (5) have been parallelized as follows:

```

1  #pragma acc parallel loop default(present) tile(32, 32)
2  for loop on image rows
3      for loop on image columns
4          #pragma acc loop
5          for loop on pixel directions
6          ...

```

Listing 3: Parallelization of collision and propagation equations.

The macroscopic state calculation for-loop of eq. (1), which requires a parallel reduction operation, has been parallelized in this way:

```

1  #pragma acc parallel loop default(present) tile(16, 16)
2  for loop on image rows
3      for loop on image columns
4          pixel_value = 0.0
5          #pragma acc loop reduction(+ : pixel_value)
6          for loop on pixel directions
7          ...

```

Listing 4: Parallelization of the calculation of the macroscopic state.

The `default(present)` directive indicates to the compiler that all variables used are present in device memory, so there's no need to make copies between host and device memory.

### 2.5.5 OBLiX framework

On one hand, the `nvc++` compiler, as well as performing code optimizations, generates code for the GPU, leading to a significant compilation time increase. This extended compilation time discourages the direct inclusion of parameters such as input images or model parameters in the source code.

On the other hand, in order to test, analyze and optimize models in a reasonable amount of time, it is necessary to be able to easily change model parameters.

To overcome this problem, the choice was made to use the OBLiX<sup>10</sup> framework. OBLiX is simple C++ library allowing both code coupling and generic execution from the input file (XML<sup>11</sup> or JSON<sup>12</sup>) containing the instructions

---

<sup>10</sup>Object Library eXecutor

<sup>11</sup>Extensible Markup Language

<sup>12</sup>JavaScript Object Notation

to be executed. Thus, OBLIX is designed to guarantee reproducibility even for complex codes in heterogenous situations.

The framework makes it possible to compile once and easily perform series of analyses by providing input files defining algorithm arguments such as input images or model parameters.

## 3 Results

Evaluate the quality and performance is rarely straightforward. Indeed, at the best of the authors' knowledge, there is no conventional metrics for dithered images. This lack of consensus lies in the complex structure resulting: dithering by definition is introducing errors to give to human eyes the illusion of a same image using less color nuances. Therefore, we used a set of different metrics that we introduce in the following section before presenting the quantitative results obtained for benchmarking the proposed methods.

### 3.1 Metrics used

#### 3.1.1 SSIM

The SSIM<sup>13</sup> is commonly used to evaluate similarities in the properties of two given images, such as the luminance (noted  $l$ ), the contrast (noted  $c$ ) and structures (noted  $s$ ). In its generic formulation, the SSIM can include these properties measurement with different weights [21]. This leads to the following expression of the SSIM between two images  $\mathcal{I}$  and  $\mathcal{J}$ :

$$\begin{aligned} \text{SSIM}(\mathcal{I}, \mathcal{J}) &= [l(\mathcal{I}, \mathcal{J})]^\alpha \cdot [c(\mathcal{I}, \mathcal{J})]^\beta \cdot [s(\mathcal{I}, \mathcal{J})]^\gamma \\ &= \left[ \frac{2\mu_{\mathcal{I}}\mu_{\mathcal{J}} + c_1}{\mu_{\mathcal{I}}^2 + \mu_{\mathcal{J}}^2 + c_1} \right]^\alpha \cdot \left[ \frac{2\sigma_{\mathcal{I}}\sigma_{\mathcal{J}} + c_2}{\sigma_{\mathcal{I}}^2 + \sigma_{\mathcal{J}}^2 + c_2} \right]^\beta \cdot \left[ \frac{\sigma_{\mathcal{I}\mathcal{J}} + c_3}{\sigma_{\mathcal{I}}\sigma_{\mathcal{J}} + c_3} \right]^\gamma, \end{aligned}$$

where  $\mu_{\mathcal{I}}$  (respectively  $\mu_{\mathcal{J}}$ ) is the mean value of the image  $\mathcal{I}$  (respectively  $\mathcal{J}$ ),  $\sigma_{\mathcal{I}}^2$  (respectively  $\sigma_{\mathcal{J}}^2$ ) is the variance of the image  $\mathcal{I}$  (respectively  $\mathcal{J}$ ), and  $\sigma_{\mathcal{I}\mathcal{J}}$  is the cross-correlation between the two images. The other values are conventional constants and weighting:  $c_1 = (k_1L)^2$ ,  $c_2 = (k_2L)^2$  and  $c_3 = \frac{c_2}{2}$  with  $L$  the dynamic range of the signal (here 255),  $k_1 = 0.01$ ,  $k_2 = 0.03$ . The weights for properties measurement are conventionally set at 1 for luminance and structure. While, contrast weight is intentionally reduced due to the inherent contrast-altering nature of the dithering process, where the range shifts from  $[0, 255]$  to  $[0, 1]$ . This results in  $\alpha = \gamma = 1$  and  $\beta = 0.1$ . It should be noted that, since the change of contrast is the same for all dithering methods, the change of weight associated is only rescaling SSIM and not affecting relative comparisons.

The SSIM is computed in two different manners: globally and locally. The GSSIM is applying the previous formula on whole images, while the MSSIM

---

<sup>13</sup>Structural Similarity Index Measure

is applying on a subset of  $11 \times 11$  pixels and taking the average value. In both cases, values closer to 1 indicate more similarity.

### 3.1.2 MLWE

The MLWE<sup>14</sup> was introduced by [22]. When the human eye is looking at an image as global set, it tends to focus on areas that seem more degraded/abnormal. For this reason the MLWE is taking into account non-linearly the maximum of error, and is computed as follows:

$$\text{MLWE}(\mathcal{I}, \mathcal{J}) = \max_{x,y} \left[ \frac{1}{25} \sum_{i=-2}^2 \sum_{j=-2}^2 \left| \frac{\mathcal{I}(x+i, y+j) - \mathcal{J}(x+i, y+j)}{\exp(0.003 W(\mathcal{I}, x+i, y+j))} \right|^2 \right]^{1/2}, \quad (20)$$

with

$$W(\mathcal{I}, x, y) = \sum_{i=-1}^1 \sum_{j=-1}^1 \left| \mathcal{I}(x+i, y+j) - \frac{1}{9} \sum_{k=-1}^1 \sum_{p=-1}^1 \mathcal{I}(x+i+k, y+j+p) \right|.$$

Thus, the lower the value of MLWE, the better quality of dithering obtained.

## 3.2 Quantitative results

The previously defined GSSIM, MSSIM, MLWE and the common MSE<sup>15</sup> metrics are computed to compare quantitatively the proposed models and usual dithering methods from literature. These comparisons are performed on several famous benchmarking images in addition with some others more adapted to dithering problems. This bundle of benchmarking images contains synthetic images and real life photographs. The compared models are those presented in section 2.3.1, section 2.3.2 and section 2.4 and noted ‘‘LBM1’’ to ‘‘LBM3’’ in the following; plus a combination of anisotropic diffusion function from section 2.3.1 with the enhanced equilibrium distribution function from section 2.4 giving a fourth model and denoted ‘‘LBM4’’ (see table 1). The proposed models are also compared with well-known ones such as models from FLOYD [2], ‘‘false FLOYD’’ [23], JARVIS [3, 24], STUCKI [4, 12] and SIERRA [23, 25].

For all the test images (Fig. 4 to Fig. 11), it is presented side by side the original image, the best (according to the metrics) of our models and the best from the ‘‘other’’ methods. For each test case, the numerical values of those metrics are presented in tables following the images (table 2 to table 9). In those tables, the bold numerical values are the best by metrics, the bold method is the one with the larger number of best metrics (so can be considered as the best method) and the symbols (b) and (c) are referring to the methods displayed in the corresponding figures. In addition to those results,

---

<sup>14</sup>Maximum Local Weighted Error

<sup>15</sup>Mean Square Error

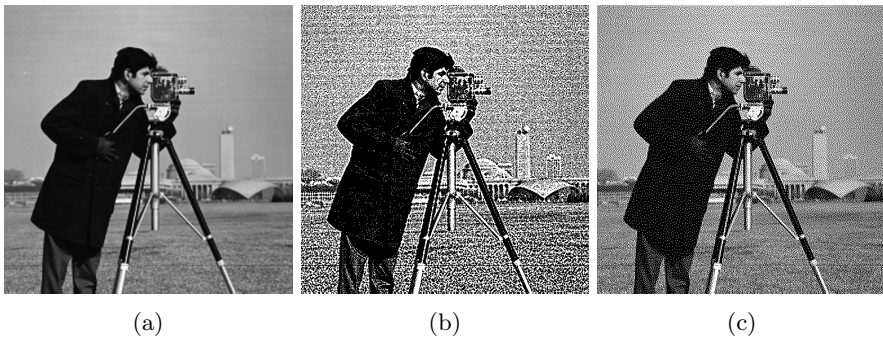


LBM model	relaxation rate	equilibrium
LBM1	eqs. (13) and (14)	eq. (8)
LBM2	eqs. (13), (15) and (16)	eq. (8)
LBM3	eq. (7) <i>i.e.</i> $\omega_i = 1$	eq. (18)
LBM4	eqs. (13) and (14)	eq. (18)

**Table 1:** Description of LBM models used.

the section A gives systematically the resulting images of our methods for all test images.

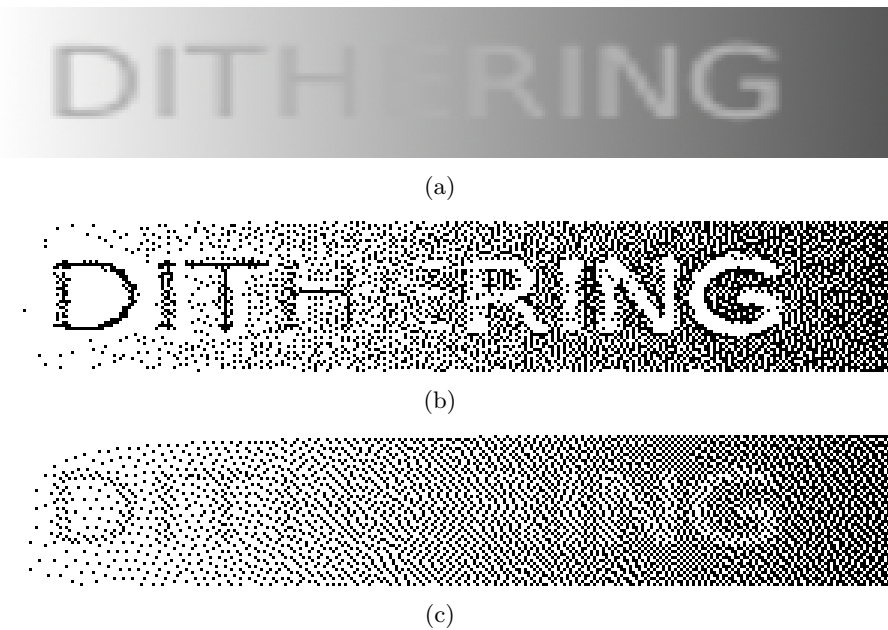
The Fig. 4 (and table 2) show dithering results on the famous cameraman image. According to table 2, the model LBM1 yields better GSSIM, MSSIM and MSE and is shown in Fig. 4(b) ; while the STUCKI’s method gives better MLWE and is printed in Fig. 4(c). It can be observed that details in the background or on the camera are more noticeable with the LBM1 method. However, an increase of the noise texture size is appearing: the spatial size of the noise “grains” is larger. On the other hand, the STUCKI’s method appears slightly more blurred.

**Fig. 4:** The cameraman test: (a) the original image, (b) the dithered image using the present model 1, (c) using the STUCKI method.

The second test image is a synthetic one. It represents the word “dithering” using a constant gray level over a gray ramp in background. This original image is illustrated in Fig. 5(a), while Fig. 5(b) is showing the dithering result with LBM1 and Fig. 5(c) with STUCKI’s method. Table 3 is clearly highlighting better results for LBM1 (and LBM2) for all metrics used over other methods and models. Despite, STUCKI’s method presents the second-best MLWE value, as it can be seen on Fig. 5(c), the letter from T to I are much less readable than on the original image or with the LBM1 method. Moreover, a small asymmetry in the restitution of the background gradient is also noticeable with STUCKI’s method (see top left corner for example).

Methods	MSSIM	GSSIM	MWLE	MSE
<b>LBM1 (b)</b>	<b>0.3287</b>	<b>0.5586</b>	37.800	<b>11 020.1</b>
LBM2	0.3201	0.5551	35.133	11 076.4
LBM3	0.2257	0.5431	33.399	11 270.8
LBM4	0.2012	0.5507	33.808	11 147.5
Floyd	0.1388	0.4814	33.888	12 269.8
FalseFloyd	0.1308	0.4799	34.099	12 295.2
Jarvis	0.1561	0.4902	33.031	12 127.2
Stucki (c)	0.1520	0.4882	<b>32.507</b>	12 159.8
Sierra	0.1542	0.4888	33.697	12 150.6

**Table 2:** Metrics results of different dithering methods applied on the cameraman image Fig. 4(a).



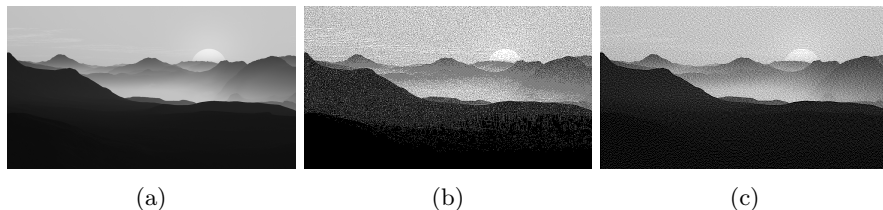
**Fig. 5:** The dithering-word test: (a) the original image, (b) the dithered image using the present model 1, (c) using the false STUCKI method.

The third test image is a higher resolution photograph of a sunset over a mountain valley. The original image (visible on Fig. 6(a)) is interesting since it mixes smooth, flatten transitions (from the fog) and sharp edges between mountains, as well as fine structures in the dark values at the foreground mountain and in the light values in the background (clouds in the sky). Table 4 gathers metrics results obtained for this image. The LBM3 model gives the best values for two out of four LBM metrics and thus can be considered as the best compared model, and is shown in Fig. 6(b). It is worth noting that LBM1 yields the best MSSIM value. But the best MLWE value is obtained with

Methods	MSSIM	GSSIM	MWLE	MSE
<b>LBM1 (b)</b>	<b>0.3454</b>	<b>0.4243</b>	<b>27.294</b>	<b>11 586.3</b>
LBM2	0.3443	0.4175	29.770	11 663.4
LBM3	0.2696	0.3834	31.766	12 049.3
LBM4	0.2543	0.3988	32.588	11 874.3
Floyd	0.1845	0.3697	32.046	12 187.1
FalseFloyd	0.1920	0.3706	30.173	12 179.7
Jarvis	0.1888	0.3770	29.717	12 077.2
Stucki (c)	0.1908	0.3761	28.345	12 089.7
Sierra	0.1876	0.3767	29.588	12 089.7

**Table 3:** Metrics results of different dithering methods applied on the “dithering-word” image Fig. 5(a).

the SIERRA’s method, as illustrated on Fig. 6(c). Comparatively, the LBM3 captures clouds in the sky and small black fluctuations in the foreground mountains; while these elements are not perceptible in SIERRA’s dithering.



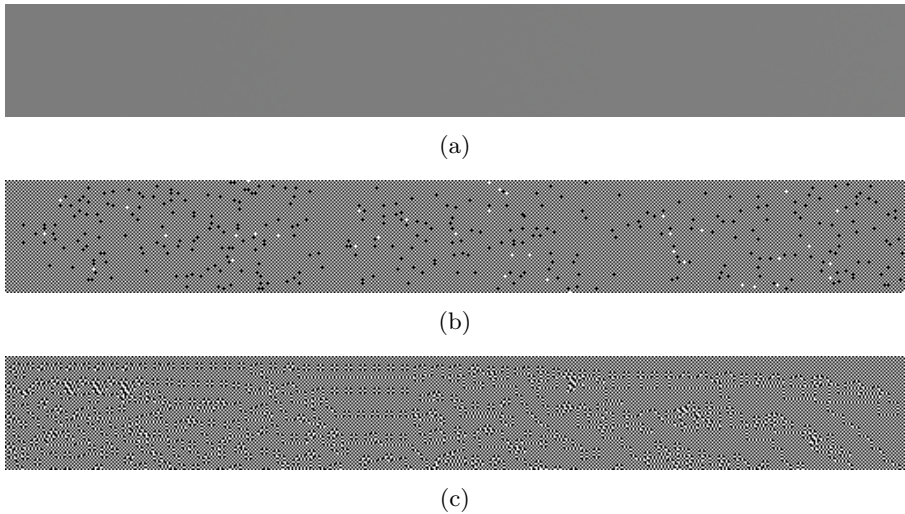
**Fig. 6:** The mountain valley test: (a) the original image, (b) the dithered image using the present model 3, (c) using the SIERRA method.

Methods	MSSIM	GSSIM	MWLE	MSE
LBM1	<b>0.3459</b>	0.6607	36.399	8713.5
LBM2	0.3457	0.6614	36.276	8697.7
<b>LBM3 (b)</b>	0.2547	<b>0.6782</b>	40.634	<b>8350.6</b>
LBM4	0.2508	0.6595	37.292	8736.9
Floyd	0.2267	0.6498	33.359	8935.4
FalseFloyd	0.2085	0.6496	33.985	8940.4
Jarvis	0.2382	0.6518	33.427	8891.7
Stucki	0.2386	0.6516	33.622	8896.0
Sierra (c)	0.2387	0.6516	<b>33.235</b>	8895.6

**Table 4:** Metrics results of different dithering methods applied on the mountain valley image Fig. 6(a).

The fourth test image is a synthetic very flat gray-gradient (ramp) starting from a value of 124 at left side to 127 at the right side (see Fig. 7(a)). Since this range of gray levels is the midpoint limit between white and black values,

it often creates artifacts or other defaults in the dithering process. Table 5 gives the metrics results, with the addition of the LEIBOVICI's entropy to track artifacts signatures. The LBM1 has the better entropy metric and STUCKI's method gives the best MLWE and is shown in Fig. 7(c). For all the other metrics the LBM4 yields better values and is shown in Fig. 7(b). The LBM4 resulting dithering is flat and without geometrical default, only black spots are present in the transition zones. By comparison, the STUCKI's results present strong geometrical defaults: relatively horizontal dashed lines appear while the transitions are vertical, plus these lines are curving downward giving the impression of a underlying structure which does not exist in the original image.

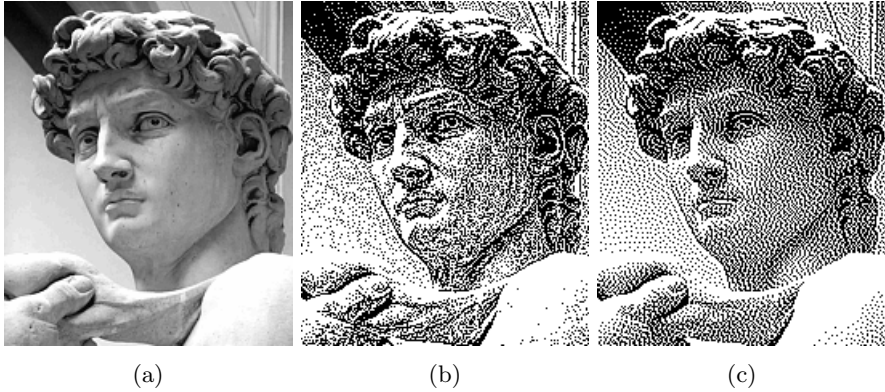


**Fig. 7:** The ramp test: (a) the original image, (b) the dithered image using the present model 4, (c) using the STUCKI method.

Methods	MSSIM	GSSIM	MWLE	MSE	Entropy
LBM1	0.4660	0.2022	4.193	16 221.8	<b>-6515.5</b>
LBM2	0.4826	0.2128	4.198	16 217.1	-9585.8
LBM3	0.4815	0.2146	4.182	16 216.3	-8377.9
<b>LBM4 (b)</b>	<b>0.4864</b>	<b>0.2174</b>	4.169	<b>16 215.1</b>	-7728.7
Floyd	0.3595	0.1378	4.138	16 250.8	-21 091.7
FalseFloyd	0.3610	0.1387	4.123	16 250.4	-20 024.7
Jarvis	0.3606	0.1403	4.116	16 248.9	-16 930.5
Stucki (c)	0.3580	0.1374	<b>4.115</b>	16 250.1	-25 606.5
Sierra	0.3584	0.1390	4.122	16 249.4	-16 697.6

**Table 5:** Metrics results of different dithering methods applied on the ramp image Fig. 7(a).

The fifth test image is the famous picture of MICHELANGELO's sculpture (see Fig. 8(a)). This test image has quite a low resolution, which is always challenging for dithering algorithms. Table 6 shows the metrics results for the different methods tested. The LBM2 has better GSSIM, MSSIM and MSE and is shown in Fig. 8(b), while JARVIS's method has better MLWE and is shown in Fig. 8(c). The LBM2 preserves finer structures of the original image with a larger noise texture size, while the JARVIS's method seems more smoothed.



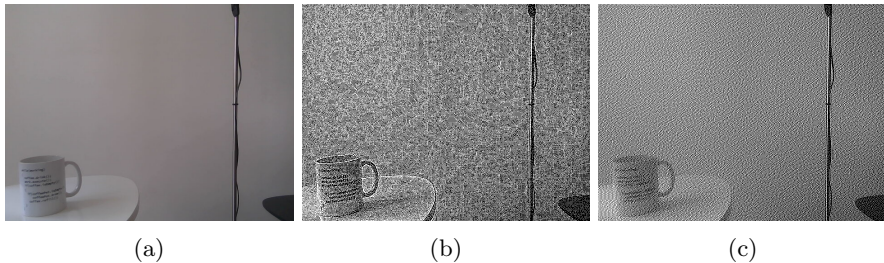
**Fig. 8:** The Michelangelo test: (a) the original image, (b) the dithered image using the present model 2, (c) using the JARVIS method.

Methods	MSSIM	GSSIM	MWLE	MSE
LBM1	0.4690	0.6127	33.471	9866.4
<b>LBM2 (b)</b>	<b>0.4711</b>	<b>0.6139</b>	30.243	<b>9845.2</b>
LBM3	0.3735	0.5853	29.892	10 339.7
LBM4	0.4436	0.6071	31.147	9962.0
Floyd	0.2477	0.5424	29.529	11 075.6
FalseFloyd	0.2393	0.5388	30.741	11 139.9
Jarvis (c)	0.3189	0.5728	<b>29.364</b>	10 545.5
Stucki	0.3012	0.5668	30.272	10 650.9
Sierra	0.3111	0.5696	32.580	10 600.5

**Table 6:** Metrics results of different dithering methods applied on the Michelangelo image Fig. 8(a).

The sixth test image is a relatively low-resolution photograph shot with a laptop camera. The image contains a mug, a table and a lamp pillar over a uniform wall (see Fig. 9(a)). The low resolution (and poor encoding) combined with the color of the wall tends to foster artifacts. Table 7 gives the metrics values computed on the test image for all compared methods. The LBM2

yields the best MSSIM value, the JARVIS's has the best MLWE result and is shown in Fig. 9(c), while the LBM1 has best GSSIM and MSE values and is given in Fig. 9(b). From the image comparison, it can be seen that the LBM1 captures more accurately the edges of the objects in the scene and the mug shadows are easily identifiable. On the other hand, JARVIS's method has less sharp edges but wall in the background seems less structured and smoother which seems more eye pleasant.



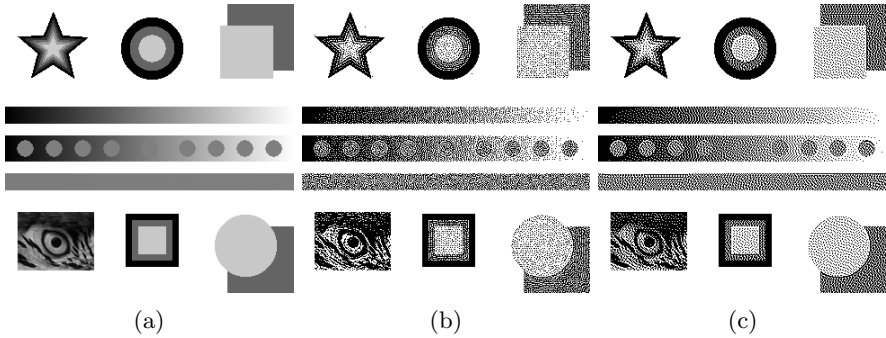
**Fig. 9:** The mug test: (a) the original image, (b) the dithered image using the present model 1, (c) using the JARVIS method.

Methods	MSSIM	GSSIM	MWLE	MSE
<b>LBM1 (b)</b>	0.4340	<b>0.1746</b>	27.813	<b>15 700.5</b>
LBM2	<b>0.4366</b>	0.1720	28.219	15 715.6
LBM3	0.3467	0.1325	31.003	15 946.4
LBM4	0.3473	0.1458	32.229	15 869.0
Floyd	0.3315	0.1530	25.748	15 826.9
FalseFloyd	0.3303	0.1512	25.635	15 837.4
Jarvis (c)	0.3365	0.1611	<b>25.608</b>	15 779.6
Stucki	0.3356	0.1590	26.332	15 791.6
Sierra	0.3367	0.1604	26.334	15 783.5

**Table 7:** Metrics results of different dithering methods applied on the mug image Fig. 9(a).

The seventh test image is a synthetic patchwork with concentric stars, circles, squares, gray-gradients (ramps) with different gray ranges with and without constant gray pellet; it also contains a real parrot eye surrounded by stripes (see Fig. 10(a)). Table 8 gathers the metrics obtained with this synthetic image. On this test, JARVIS's method can be considered the best method with 3 best metrics (GSSIM, MLWE, MSE) out of 4, and is shown in Fig. 10(c). The LBM2 yields the best MSSIM value and is illustrated in Fig. 10(b). Despite, the geometrical default in gradients or the top right square with the JARVIS's method, the edges are clear and well captured. With the LBM2, small artifacts remains in the gradient and some black pixels close to

the star are infidels to the original image. One can also notice the residues of the diffusion process in the onion skin layers in the dark squares in the top and bottom right part of the image.

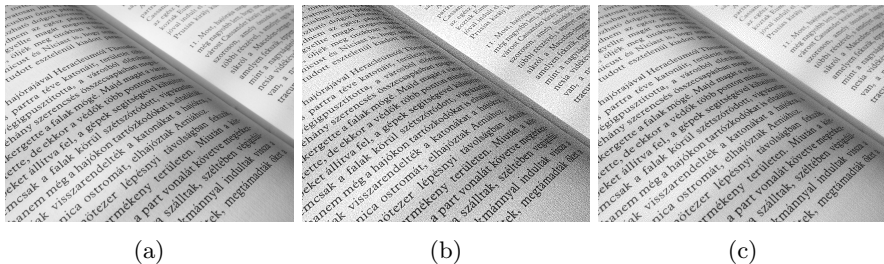


**Fig. 10:** The synthetic patchwork test: (a) the original image, (b) the dithered image using the present model 2, (c) using the JARVIS method.

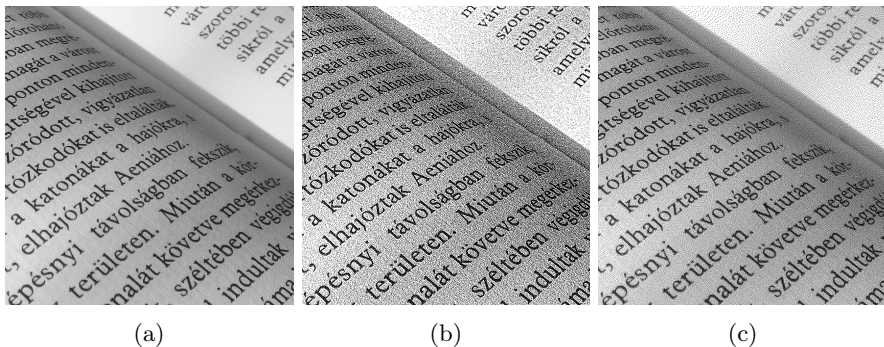
Methods	MSSIM	GSSIM	MWLE	MSE
LBM1	0.6840	0.7402	35.079	5550.2
LBM2 (b)	<b>0.6966</b>	0.7484	31.855	5399.4
LBM3	0.6809	0.7411	33.012	5535.4
LBM4	0.6788	0.7375	47.510	5600.6
Floyd	0.6765	0.7426	34.752	5506.7
FalseFloyd	0.6741	0.7418	30.078	5521.3
<b>Jarvis (c)</b>	0.6829	<b>0.7489</b>	<b>29.705</b>	<b>5391.7</b>
Stucki	0.6823	0.7478	29.709	5411.5
Sierra	0.6822	0.7484	29.262	5401.7

**Table 8:** Metrics results of different dithering methods applied on the synthetic patchwork image Fig. 10(a).

The eighth and last test image is a photograph of book pages with a relatively high resolution, see Fig. 11(a). It is interesting to benchmark the restitution of fine structures as letters and the shading visible on the page when the resolution is larger. Table 9 gives the numerical values obtained for the considered metrics with all tested methods. The LBM1 yields better value of MSSIM, GSSIM and MSE and is shown in Fig. 11(b), while the false FLOYD's algorithm presents better MLWE value and is printed in Fig. 11(c). From a human eye's point of view, looking at the picture globally, the differences between the resulting images and the original one are almost imperceptible. When looking closer at detailed structures (see zoomed pictures in Fig. 12), the LBM1 results are capturing structures sharply while false FLOYD's method is a little bit more blurred.



**Fig. 11:** The textbook test: (a) the original image, (b) the dithered image using the present model 2, (c) using the false FLOYD method.



**Fig. 12:** The textbook test zoom: (a) the original image, (b) the dithered image using the present model 2, (c) using the false FLOYD method.

### 3.3 Computation time

To compare the computation time of the proposed models, a comparison has been proceeded on the largest resolution test image (textbook image, see Fig. 11(a)) with different acceleration options. These accelerations are enabled simply thanks to the OPENACC meta-commands. Thus, a sequential execution, a multi-CPU<sup>16</sup> (multi-thread) parallelization and a GPU parallelization, was tested. The results were obtained on a laptop using: 11<sup>th</sup> Generation Intel Core i9 @ 2.60 GHz  $\times$  16, 32.0 GB of RAM<sup>17</sup> and a NVIDIA RTX A2000 Mobile (4 GB - Ampere 86). Table 10 gives the computation time obtained, with the different methods and execution strategies.

On the one hand, for all the LBM models, a factor 3 is gained when using CPU parallelization compared to sequential execution, and another extra factor 3 is saved with GPU parallelization. These gains are those expected with such techniques, and show the good scalability of the LBM. On the other hand, the other models tested here, that can be qualified as “filter” algorithms,

<sup>16</sup>Central Processing Unit

<sup>17</sup>Random-Access Memory



Methods	MSSIM	GSSIM	MWLE	MSE
<b>LBM1 (b)</b>	<b>0.4297</b>	<b>0.5420</b>	37.620	<b>9577.8</b>
LBM2	0.4258	0.5400	40.326	9601.6
LBM3	0.3633	0.5127	36.226	9933.7
LBM4	0.3531	0.5273	38.459	9756.1
Floyd	0.1668	0.4312	32.750	10924.3
FalseFloyd (c)	0.1656	0.4276	<b>32.628</b>	10969.3
Jarvis	0.2059	0.4516	33.191	10673.4
Stucki	0.1965	0.4475	32.952	10723.5
Sierra	0.2007	0.4485	33.048	10710.7

**Table 9:** Metrics results of different dithering methods applied on the textbook image Fig. 11(a).

Methods	seq (s)	Mutli-CPU (s)	GPU (s)
LBM1	291.550	90.359	25.392
LBM2	252.557	86.537	24.201
LBM3	134.051	42.109	11.187
LBM4	340.378	106.289	28.447
Floyd	0.108	0.125	0.120
FalseFloyd	0.076	0.088	0.086
Jarvis	0.177	0.197	0.189
Stucki	0.170	0.198	0.193
Sierra	0.170	0.196	0.189

**Table 10:** Computation time of different dithering methods applied on the textbook image Fig. 11(a) for difference acceleration strategy.

need browsing through the image, one pixel after another. And the final result is available once the rolling shutter has covered the whole image. This simple scan through makes the execution very fast, since there is no time evolution, nor diffusive process, nor converging to be reached. One can notice that our sequential implementation of “filter” models seem un-scaling. However, for many years it is possible to split the winding path of dithering filters to have parallel versions of these algorithms [26–28], leading to improved computational times. Yet, our naive sequential implementation does not change the fact that these “filter” methods are clearly less computational time expansive: the computational time ratio with the LBM models is around 100.

## 4 Discussion

The principle of using LBM for performing dithering operations is not new. In their seminal work, HAGENBURG *et al.* chose to use a unitary  $\omega$  parameter, which has the effect of generating a uniform and isotropic dithering operation. In this paper, other values of  $\omega$  are tested, especially values between 0.1 and 1. The relaxation time value of 0.35 seemed optimal in the proposed implementations. However, the dithering algorithm remains uniform and isotropic in this case.

Our contribution lies in the development of models using variable relaxation rates and equilibrium functions based on the image characteristics. This allows for consideration of image anisotropy in the algorithm. In particular, we proposed four models using different relaxation rate functions and equilibrium functions. These relaxation rate functions are based on a general formulation (given in eqs. (12) and (13)), parameter sets (see eqs. (14) and (15)), which involve a three-level treatment depending on the values of the considered gray levels.

Besides, the new equilibrium distribution function proposed in eq. (18) allows reducing artifacts. GPU implementation was also carried out to achieve stronger performances. In particular, collision and propagation steps as well as the calculation of macroscopic states were parallelized. This parallelization effort has been made to emphasize an argument of parallel efficient which is often given about the LBM. Such efficiency is coming from the “local” character of the method. However, our contribution is not aiming any performance or optimized algorithms. Indeed, we illustrate that argument by showing with little effort our LBM models can be easily parallelized on GPU and therefore are providing better results. Nevertheless, these performances are far from well optimized commercial or dedicated LBM software.

While, the computational efficiency of the LBM is undeniable in fluid dynamics, it should be considered with caution when applied to image processing. Indeed, the LBM is categorized as a local method: all but streaming operations could be performed independently in parallel; so it scales very well. However, it is related to a time evolution process (such as diffusion), which is not necessarily fast. Whereas image processing is often leaning on well-optimized techniques, as is the case for dithering, that are not relying on time evolution processing. In such situations, the LBM offers an advantage, providing PDE<sup>18</sup> modelling the image processing which enable deeper understanding, but also offers a disadvantage in computational time. This disadvantage can generally be explained by the relatively slow time evolution process (intrinsic of the method). Even though it is not the aim of our work, such slow time evolution and “locality” can be traded for steady-state algorithm resulting in an improved computational time, as it has been illustrated in [29]. This, also, explains why the computational time of our LBM models which require several hundred of time steps to reach a converged state is not comparable with filter models that require only one time step. The fact our work tends to a steady-state diffusion regime could also play a role in its relative better quality results.

Qualitatively, several phenomena can be observed in the results, corresponding to different test images. First, a reduction in artifacts in the proposed models can be observed compared to the HAGENBURG algorithm. Fig. 1 shows the appearance of these artifacts for geometric shapes, especially in the case of  $\omega = 1$ . Next, the presented methods seem to produce noisier results than

---

<sup>18</sup>Partial Differential Equations

other methods on flat areas and simple geometric shapes in the images. However, areas with tortuosity and more natural and complex shapes appear to be better preserved. It is also noticeable that the contrast appears slightly improved, and the “fog” effect that can be obtained with isotropic dithering methods such as the FLOYD and STUCKI algorithms is lost. In fact, with the proposed methods, the algorithm implicitly induces a sharpening effect. Lastly, the characteristics of the methods are well summarized in Fig. 5, showing a general increase in noise, a slight increase in contrast, and the preservation of contours, which makes the word “dithering” more readable than in the STUCKI algorithm.

For quantitative measures, the choice of commonly used metrics has been made to evaluate dithering algorithms, namely MSSIM, GSSIM, MLWE, and MSE. On most images, the metrics show better results for either of the developed models. It seems that the weakness of our models lies in the increase in noise in very dark areas of the image, which leads to higher MLWE values. This is most likely related to the overall increase in noise.

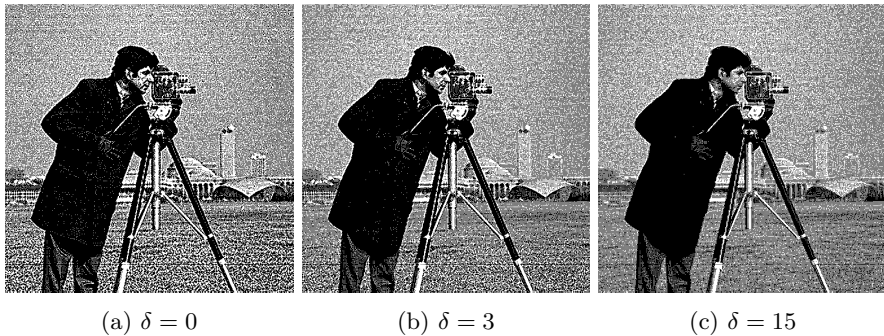
Among the most important limitations, one can mention the critical size of structures. Indeed, the LBM scheme is defined in  $D2Q9$ , which implies a certain preferred structure size, as in wavelet analysis. This phenomenon can be observed by comparing Fig. 8 (low resolution) and 11 (high resolution): it is observed that when the resolution is higher, the increase in noise is much less significant.

Despite the optimizations, computation times of the methods presented in this paper remain longer than for classical isotropic algorithms. While this can be counted as a limitation, it should be considered in the light of the contribution of structure preservation. It should also be noted that no specific algorithm optimization nor sensitivity analysis in regard to the number of iterations, has been performed.

In the light of the quantitative results obtained on the different test-cases realized, no LBM model can be considered as a universal best quality yielder. Yet in a method selection process it should consider that: if all LBM models are yielding better metrics, due to their computational time they should be preferred only when the quality is at stake. When this is the case, despite that model and parameters selection is debatable (since it is case-specific), the LBM3 model can be considered in a first intention, due to its quality over computational time ratio. While, if the computational time is aimed, then one should prefer parallel “filter” dithering models [26–28].

A future work will focus on the development of algorithms based on LBM schemes larger than the classical  $D2Q9$ , with the belief that expanding this scheme will overcome the critical structure size issue. In addition, reflections on an extension to multi-toning or color dithering could be carried out. Finally, better parameter tuning or optimization could potentially improve quantitative performance. For example, regarding the  $\delta$  parameter, it can be seen on Fig. 13 that a lower  $\delta$  value leads to larger pixel clusters but also to high contrast and finer structure preserved, whereas a high value tends to make the

pixel clusters disappear in favor of a grid pattern, making the image smoother. The computational time of LBM models can be reduced by at least 4 when decreasing the parameter  $t_{\text{limit}}$  in eq. (13), but that in return, reduce the quality of results. Therefore, an optimal trade off remains to be determined. Since these heuristics is based on observations and the scrutiny of quality metrics, theoretical proof of convergence, correctness or complexity would lead to a deep improvement and better understanding of the dithering processing.



**Fig. 13:** The cameraman dithered with LBM4 using different  $\delta$  values

## 5 Conclusions

This paper proposes a generalization of dithering methods using the LBM. By taking into consideration the anisotropy of the image in the expression of the relaxation rate via the integration of the image gradient in its expression, or by adapting the equilibrium distribution function, it has been possible to establish different dithering models which adapt to the image structures.

The four proposed models have different characteristics that lead to specific performances. Even if the proposed methods present larger noise patterns than other methods, this is compensated by better preservation of structures, and, therefore, better preservation of the information contained in the image.

The qualitative results illustrate these phenomena. The quantitative results, measured using metrics classically used in the field of dithering, are almost always better with the proposed methods on the test images used. Future work will focus on optimizing model parameters, in general or per image, in particular with regard to residual noise. Until further optimization, the presented algorithms, despite a great scalability and better quantitative results, remain more computationally time demanding. Therefore, they might be more appropriate to complicated cases where quality prevails. This will improve the results both qualitatively and quantitatively.

**Acknowledgments.** The authors would like to thank reviewers for their enlightening remarks that helped improving this work.

## Declarations

- Funding: Pulsar - Pays de la Loire n° 2022\_09765.
- Conflict of interest/Competing interests: None
- Ethics approval: 'Not applicable'
- Consent to participate: 'Not applicable'
- Consent for publication: 'Not applicable'
- Code availability: upon reasonable request.
- Authors' contributions: Conceptualization: Romain NOËL (RN), Alexis RENIER-ROBIN (ARR), Laurent NAVARRO (LN). Methodology: RN, ARR, LN. Implementation: RN, ARR. Data curation: RN, ARR. Visualization: RN. Writing: RN, ARR, LN. Revision and edition: RN, ARR, LN. All authors agreed to the published version of the manuscript.

## Appendix A Graphical results of our models

### Abbreviations

*BGK* BHATNAGAR, GROSS and KROOK 4

*CPU* Central Processing Unit 23

*GCC* GNU Compiler Collection 12

*GPU* Graphics Processing Unit 3, 11, 13, 23, 25

*GSSIM* Global Structural Similarity Index Measure 3, 6, 7, 14–16, 20–22, 26

*HPC* High Performance Computing 12

*JSON* JavaScript Object Notation 13

*LBM* Lattice BOLTZMANN Method 2–4, 8, 11–13, 23–27

*MLWE* Maximum Local Weighted Error 15–17, 19–22, 26

*MSE* Mean Square Error 15, 16, 20–22, 26

*MSSIM* Mean Structural Similarity Index Measure 3, 14–17, 20–22, 26

*OPENACC* Open ACCelerators 11–13, 23

*OBLIX* OBject Library eXecutor 13, 14

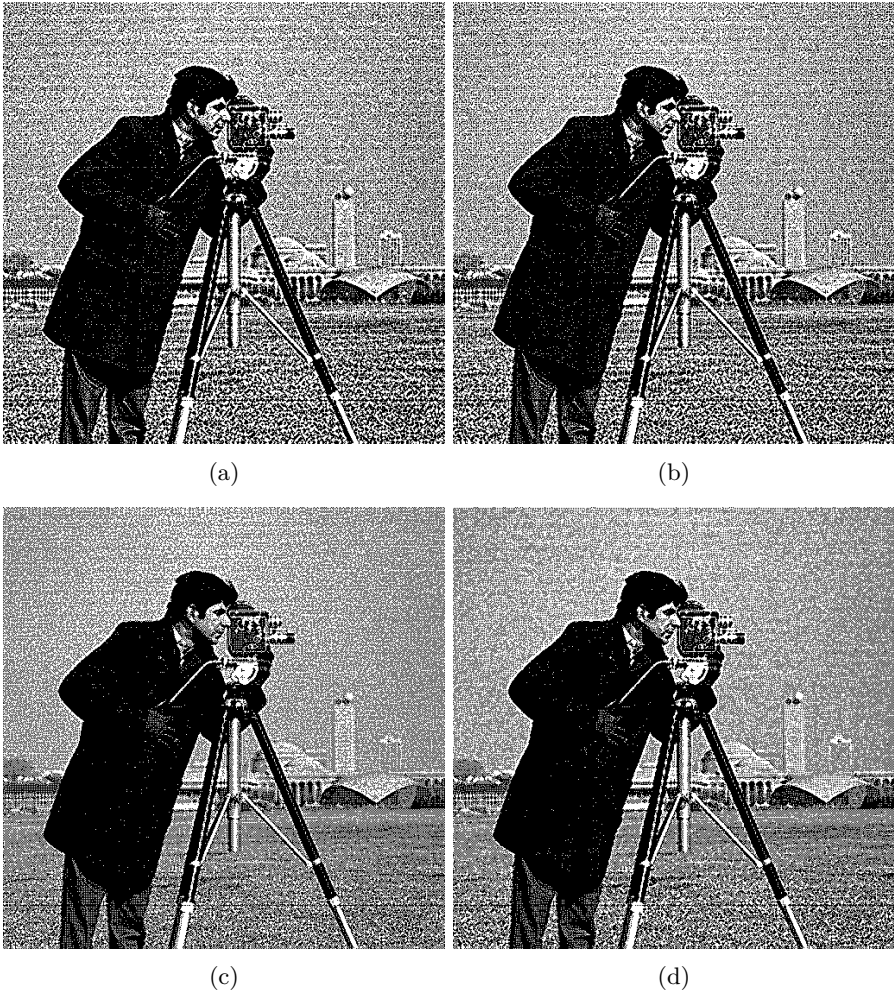
*PDE* Partial Differential Equations 25

*RAM* Random-Access Memory 23

*SDK* Software Development Toolkit 12

*SSIM* Structural Similarity Index Measure 14

*XML* Extensible Markup Language 13



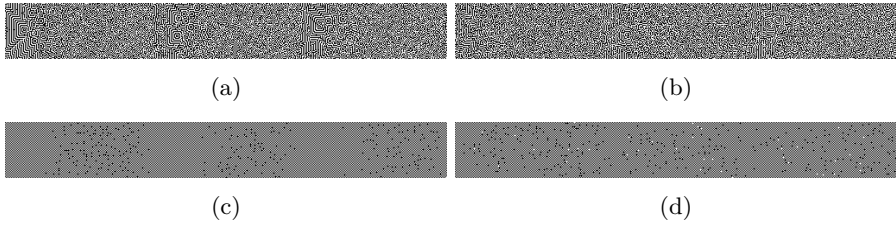
**Fig. A1:** The dithering results on the cameraman case with our model 1 (a), model 2 (b), model 3 (c) and model 4 (d).

## Nomenclature

- $\alpha$  minimum equilibrium threshold 10, 11
- $\beta$  maximum equilibrium threshold 10, 11
- $\delta$  offset equilibrium value 10, 11, 26, 27
- $\mathcal{I}$  image function of  $\mathbb{R}^D$  in  $\mathbb{R}$  14, 15
- $\mathcal{J}$  image function of  $\mathbb{R}^D$  in  $\mathbb{R}$  14, 15
- $\mu$  mean value 14
- $\nu$  minimum equilibrium acceptance value 5, 10
- $\sigma$  cross-correlation between two signals 14



**Fig. A2:** The dithering results on the dithering-word case with our model 1 (a), model 2 (b), model 3 (c) and model 4 (d).



**Fig. A3:** The dithering results on the gradient (ramp) case with our model 1 (a), model 2 (b), model 3 (c) and model 4 (d).

$\sigma$  standard deviation 14

$i$  index discretized distribution 4, 5, 7–10, 16

$c$  contrast metric 14

$l$  luminance metric 14

$s$  structure metric 14

$\Omega(.,.)$  collision operator 4, 7

$\rho$  mass density 4, 5, 7, 8, 10–12

$D$  physical space dimension 4, 5, 26

$e_i$  discretized space stencil 4, 5, 7, 8, 10, 11

$f$  density distribution function over velocity space 7

$f^{eq}$  equilibrium distribution 7

$f_i^{eq}$  discretized equilibrium distribution 4, 5, 7, 10

$f_i$  discretized density distribution 4, 7

$\omega$  relaxation frequency 4–10, 16, 24, 25

$\Delta t$  increment of time 4, 7, 11

$\psi$  Model 2 random variable 10

- $\mathbf{x}$  coordinate Eulerian vector field 4, 5, 7–12
- $Q$  number of discretized speed used in the lattice 4, 5, 26
- $g_i$  function  $g_i$  of rate function 8, 9
- $a$  parameter 1 of function of rate function 8, 9
- $b$  parameter 2 of function of rate function 8, 9
- $c$  parameter 3 of function of rate function 8, 9
- $d$  parameter 4 of function of rate function 8, 9
- $t$  time 4, 5, 7–12
- $\mathcal{U}$  uniform law symbol 10
- $t_i$  discrete weight of the GAUSS quadrature 5, 7, 10
- $\xi_i$  discretized particles velocity 4

## References

- [1] Yellott Jr, J.I.: Spectral consequences of photoreceptor sampling in the rhesus retina. *Science* **221**(4608), 382–385 (1983)
- [2] Floyd, R., Steinberg, L.: An adaptive algorithm for spatial grey scale. In: *SID Digest*, 1975, vol. 17, pp. 36–37 (1975)
- [3] Jarvis, J., Roberts, C.: A New Technique for Displaying Continuous Tone Images on a Bilevel Display. *IEEE Transactions on Communications* **24**(8), 891–898 (1976). <https://doi.org/10.1109/TCOM.1976.1093397>
- [4] Stucki, P.: Image processing for document reproduction. In: Stucki, P. (ed.) *Advances in Digital Image Processing*, pp. 177–218. Springer US, Boston, MA (1979). [https://doi.org/10.1007/978-1-4615-8282-3\\_9](https://doi.org/10.1007/978-1-4615-8282-3_9)
- [5] Shiau, J.-N., Fan, Z.: Set of easily implementable coefficients in error diffusion with reduced worm artifacts. In: *Color Imaging: Device-Independent Color, Color Hard Copy, and Graphic Arts*, vol. 2658, pp. 222–225 (1996). SPIE
- [6] Ostromoukhov, V.: A simple and efficient error-diffusion algorithm. In: Pockock, L. (ed.) *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '01*, vol. 1778, pp. 567–572. Association for Computing Machinery, New York, NY, USA (2001). <https://doi.org/10.1145/383259.383326>
- [7] Bobylev, A.V.: The Chapman-Enskog and Grad methods for solving the Boltzmann equation. In: *Akademiia Nauk SSSR Doklady*, vol. 262, pp. 71–75 (1982)



- [8] Chang, Q., Yang, T.: A lattice Boltzmann method for image denoising. *IEEE Transactions on Image Processing* **18**(12), 2797–2802 (2009). <https://doi.org/10.1109/TIP.2009.2028369>
- [9] Chen, Y., Yan, Z., Shi, J.: Application of lattice Boltzmann method to image segmentation. In: 2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 6561–6564 (2007). IEEE
- [10] Hagenburg, K., Breuß, M., Vogel, O., Weickert, J., Welk, M.: A lattice Boltzmann model for rotationally invariant dithering. In: *Advances in Visual Computing: 5th International Symposium, ISVC 2009, Las Vegas, NV, USA, November 30-December 2, 2009. Proceedings, Part II* 5, pp. 949–959 (2009). Springer
- [11] Hu, X.Y.: Simple gradient-based error-diffusion method. *Journal of Electronic Imaging* **25**(4), 043029 (2016). <https://doi.org/10.1117/1.JEI.25.4.043029>
- [12] Stucki, P.: MECCA-A multiple-error correction computation algorithm for bi-level image hardcopy reproduction. IBM, Watson Research Center, Ossining, NY, USA (1991)
- [13] Navarro, L., Courbebaisse, G., Roux, C.: La Méthode Lattice Boltzmann en Traitement D’image. In: *TAIMA 2015, Traitement et Analyse de l’Information - Méthodes et Applications*, Hammamet, Tunisie (2015)
- [14] Leibovici, D.G.: Defining spatial entropy from multivariate distributions of co-occurrences. In: Hornsby, K.S., Claramunt, C., Denis, M., Ligozat, G. (eds.) *Spatial Information Theory* vol. 5756, pp. 392–404. Springer, Berlin, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-03832-7\\_24](https://doi.org/10.1007/978-3-642-03832-7_24)
- [15] Bauer, M., Köstler, H., Rüde, U.: lbmpy: Automatic code generation for efficient parallel lattice boltzmann methods. *Journal of Computational Science* **49**, 101269 (2021). <https://doi.org/10.1016/j.jocs.2020.101269>
- [16] Januszewski, M., Kostur, M.: Sailfish: A flexible multi-gpu implementation of the lattice boltzmann method. *Computer Physics Communications* **185**(9), 2350–2368 (2014). <https://doi.org/10.1016/j.cpc.2014.04.018>
- [17] Latt, J., Malaspinas, O., Kontaxakis, D., Parmigiani, A., Lagrava, D., Brogi, F., Belgacem, M.B., Thorimbert, Y., Leclaire, S., Li, S., Marson, F., Lemus, J., Kotsalos, C., Conradin, R., Coreixas, C., Petkantchin, R., Raynaud, F., Beny, J., Chopard, B.: Palabos: Parallel lattice boltzmann solver. *Computers & Mathematics with Applications* **81**, 334–350 (2021). <https://doi.org/10.1016/j.camwa.2020.03.022>. Development and

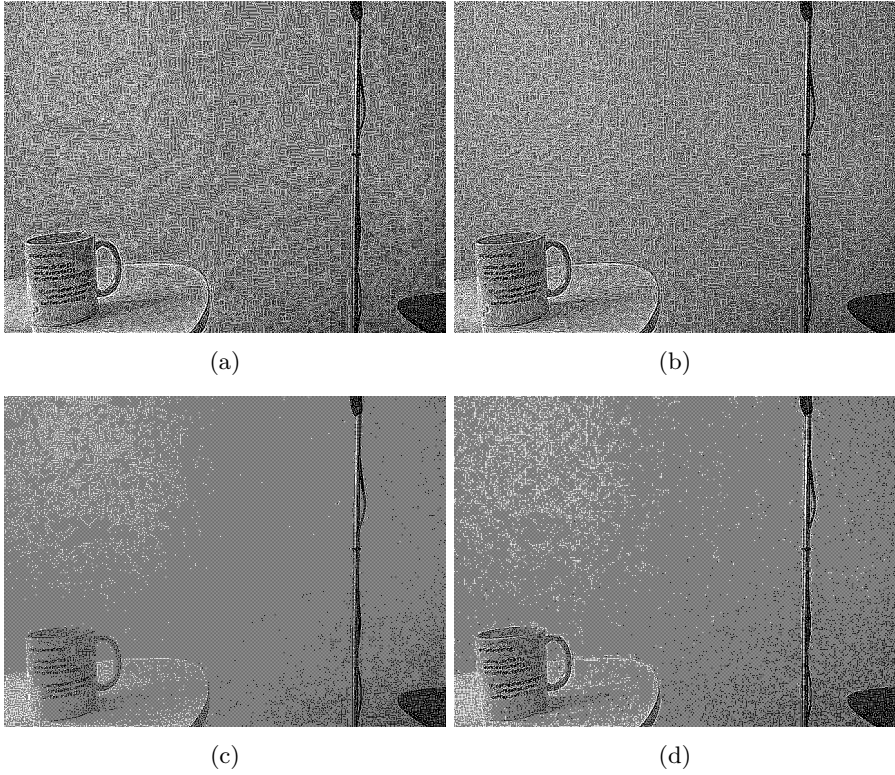
## Application of Open-source Software for Problems with Numerical PDEs

- [18] Tomczak, T., Szafran, R.G.: A new gpu implementation for lattice-boltzmann simulations on sparse geometries. *Computer Physics Communications* **235**, 258–278 (2019). <https://doi.org/10.1016/j.cpc.2018.04.031>
- [19] Bernaschi, M., Fatica, M., Melchionna, S., Succi, S., Kaxiras, E.: A flexible high-performance lattice boltzmann gpu code for the simulations of fluid flows in complex geometries. *Concurrency and Computation: Practice and Experience* **22**(1), 1–14 (2010) <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.1466>. <https://doi.org/10.1002/cpe.1466>
- [20] Wittmann, M., Haag, V., Zeiser, T., Köstler, H., Wellein, G.: Lattice boltzmann benchmark kernels as a testbed for performance analysis. *Computers & Fluids* **172**, 582–592 (2018). <https://doi.org/10.1016/j.compfluid.2018.03.030>
- [21] Wang, Z., Simoncelli, E.P., Bovik, A.C.: Multiscale structural similarity for image quality assessment. In: *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, pp. 1398–14022 (2003). <https://doi.org/10.1109/ACSSC.2003.1292216>
- [22] Mitsa, T.: Image quality metrics for halftone images. In: Heaston, R.J. (ed.) *Imaging Technologies and Applications*, vol. 1778, pp. 196–207. SPIE, Chicago, IL (1992). <https://doi.org/10.1117/12.130974>
- [23] Crocker, L.D., Boulay, P., Morra, M.: *Digital Halftoning or Dithering* (1991)
- [24] Jarvis, J.F., Judice, C.N., Ninke, W.H.: A survey of techniques for the display of continuous tone pictures on bilevel displays. *Computer Graphics and Image Processing* **5**(1), 13–40 (1976). [https://doi.org/10.1016/S0146-664X\(76\)80003-2](https://doi.org/10.1016/S0146-664X(76)80003-2)
- [25] Sierra, F. *CIS Graphics Support Forum* (1989)
- [26] Metaxas, P.T.: Optimal parallel error diffusion dithering. In: Beretta, G.B., Eschbach, R. (eds.) *Electronic Imaging '99*, San Jose, CA, pp. 485–494 (1998). <https://doi.org/10.1117/12.334593>
- [27] Deshpande, A., Misra, I., Narayanan, P.J.: Hybrid implementation of error diffusion dithering. In: *2011 18th International Conference on High Performance Computing*, pp. 1–10. IEEE, Bengaluru, India (2011). <https://doi.org/10.1109/HiPC.2011.6152714>

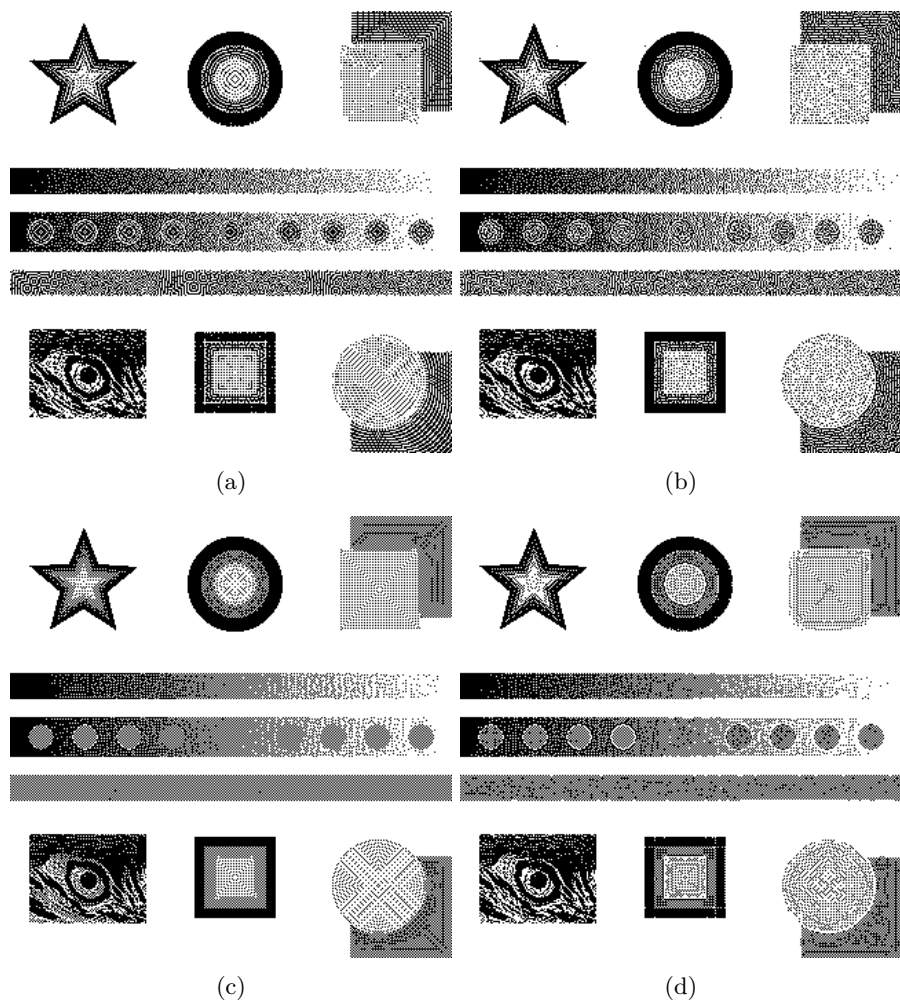
- [28] Zhang, Y., Recker, J.L., Ulichney, R., Beretta, G.B., Tastl, I., Lin, I.-J., Owens, J.D.: A parallel error diffusion implementation on a GPU. In: Owens, J.D., Lin, I.-J., Zhang, Y.-J., Beretta, G.B. (eds.) IS&T/SPIE Electronic Imaging, San Francisco Airport, California, USA, p. 78720 (2011). <https://doi.org/10.1117/12.872616>
- [29] Ginzburg, I.: Steady-state two-relaxation-time lattice Boltzmann formulation for transport and flow, closed with the compact multi-reflection boundary and interface-conjugate schemes. *Journal of Computational Science* **54**, 101215 (2021). <https://doi.org/10.1016/j.jocs.2020.101215>



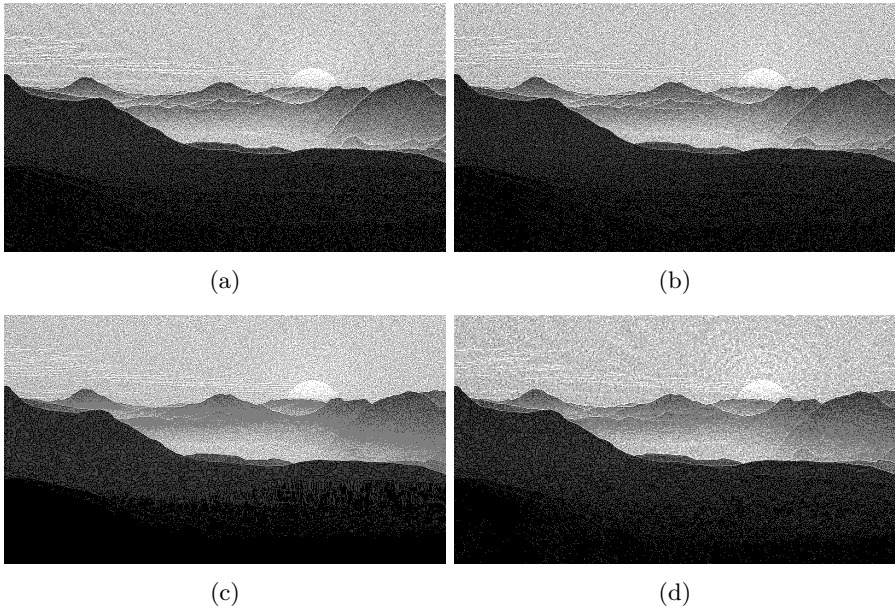
**Fig. A4:** The dithering results on the michelangelo case with our model 1 (a), model 2 (b), model 3 (c) and model 4 (d).



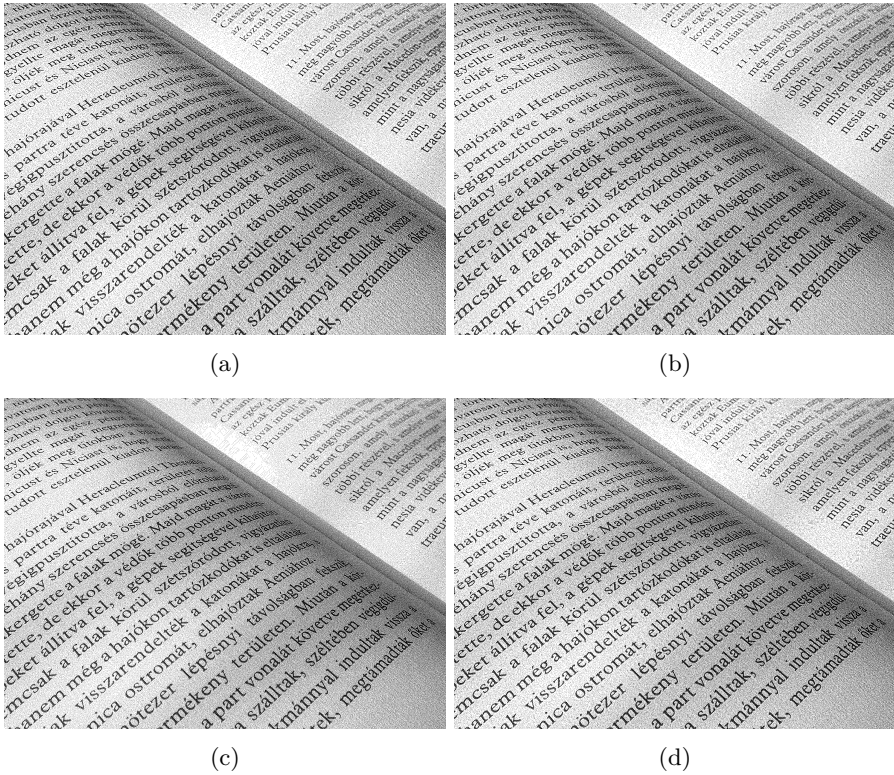
**Fig. A5:** The dithering results on the mug case with our model 1 (a), model 2 (b), model 3 (c) and model 4 (d).



**Fig. A6:** The dithering results on the synthetic case with our model 1 (a), model 2 (b), model 3 (c) and model 4 (d).



**Fig. A7:** The dithering results on the gallery case with our model 1 (a), model 2 (b), model 3 (c) and model 4 (d).



**Fig. A8:** The dithering results on the textbook case with our model 1 (a), model 2 (b), model 3 (c) and model 4 (d).