AIMS Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 7(2): 2176–2194.
DOI:10.3934/math.2022124
Received: 01 September 2021
Accepted: 14 October 2021
Published: 09 November 2021

Research article

Modified Poisson compositing technique on skewed grid

Nordin Saad[1], **A'qilah Ahmad Dahalan**[2] **and Azali Saudi**[1,*]

[1] Faculty of Computing and Informatics, Universiti Malaysia Sabah, Kota Kinabalu, Malaysia
[2] Department of Mathematics, Universiti Pertahanan Nasional Malaysia, Kuala Lumpur, Malaysia

* **Correspondence:** Email: azali@ums.edu.my.

Abstract: Image compositing is the process of seamlessly inserting a portion of a source image into a target image to create a new desirable image. This work describes an image composition approach based on numerical differentiation utilizing the Laplacian operator. The suggested procedure uses the red-black strategy to speed up computations by using two separate relaxation factors for red and black nodes, as well as two accelerated parameters on a skewed grid. The Skewed Modified Two-Parameter Overrelaxation (SkMTOR) approach is a modification of the existing MTOR method. The SkMTOR has been used to solve numerous linear equations in the past, but its applicability in image processing has never been investigated. Several examples were used to test the suggested method in solving the Poisson equation for image composition. The results demonstrated that the image composition was successfully constructed using all six methods considered in this study. The six methods evaluated yielded identical images based on the similarity measurement results. In terms of computing speed, the skewed variants perform much quicker than their corresponding regular grid variants, with the SkMTOR showing the best performance.

Keywords: image compositing; Poisson equation; finite difference method; modified iterative method; MSOR; MAOR; MTOR; skewed grid; image blending; composition
Mathematics Subject Classification: 35J05, 65N06, 94A08

1. Introduction

In image processing, image composing, as an interactive image editing procedure, plays a critical role. Its goal is to create a new image by naturally and seamlessly combining selected image patches from one or more images into one base image. It is commonly used in film making, photo editing, and web design, among other things. Image matting [1], image stitching [2, 3], surface reconstruction [4], image completion [5], video blending [6], and image inpainting [7] are examples of other image processing techniques that use a similar approach.

In [8], Poisson equation was used to model the image composition process, in which the image gradients were computed with the Laplacian operator. The gradient field inside the cloned region was obtained from the source image, and the Dirichlet boundary conditions were determined by the cloned region where the pixel colour was from the target image. Since then, several works had been conducted in applying image gradients for image processing. Raskar et al. [9] used a method that preserves local perceptual qualities while reducing the effects of aliasing, haloing, and ghosting. Sengupta et al. [10] suggested a method for solving the Poisson equation in the matte gradient domain to solve the natural image matting problem. Their research showed that the proposed method offered promising results outcomes for a variety of natural images. Wang & Yang [2] then developed an image stitching method for combining multiple images with some overlapped images. The similarity between the photos and the visibility of the seam were built in the gradient domain. The authors demonstrated that the proposed strategy was capable of eliminating global inconsistencies and limiting difficulties caused by the illumination effect by working in the gradient domain rather than the pixel intensities of the images. A completion approach based on the gradient domain that used a patch-based technique was presented in [5]. The study was divided into two phases: first, they used gradient maps to fill in the missing area using a patch-based filling method; and second, they used the gradient maps to fill in the missing area using a patch-based filling algorithm. By solving the Poisson equation, the image was rebuilt from the gradient maps. A matching criterion based on the gradient and colour was presented to obtain better image completion.

The study in [4] for surface reconstruction using an algebraic technique also included image editing based on the gradient domain. They conducted a thorough examination of several integration strategies as well as the difficulty of working with noisy gradient fields. Similar approaches were proposed in [11] and [12]. The work in [13] presented an image composition technique for reducing bleeding artefacts in generated images. In [14] the gradient-based approach was applied for video editing. A Fourier implementation to solve the Poisson equation for image processing was reported in [15]. The Poisson equation is solved in the Retinex algorithm by Limare et al. [16] using Neumann boundary conditions and Fourier implementation. To remove the traces left by camera anonymization used for privacy protection and anti-tracking, Poisson blending was utilized in [17]. The suggested strategy efficiently suppresses the visual artefacts produced by camera anonymization while preserving anti-forensic efficiency. Hussain & Kamel [18] proposed the image pyramid method for solving the Poisson equation. In their work on image blending, Afifi & Hussain [19] suggested the modified Poisson solver. [20] and [21] proposed several iterative techniques to obtain the solutions of the Poisson equation for image editing purposes.

More recently, Cao et al. [22] proposed a self-embedding image watermarking scheme based on reference sharing and the Poisson equation. In their work, they established the relationship between each pixel and its neighborhood in the original image via the Laplacian operator and converted it to compression bits. The approach increased the recovered area from the tampered image by reconstructing the relationship between each compression bit and each reference bit. The work in [23] used the Poisson blending technique for cloud removal algorithm. A quad-tree approach was utilized to speed up the proposed gradient-domain compositing method. Experimental photos on a large area of Landsat data with less than 80% cloud coverage yielded encouraging results. Pan et al. [24] suggested a method that combines a deep convolutional generative adversarial network and Poisson blending to solve image completion tasks in their work to obtain complete total electron content maps used

in global navigation satellite system. In [25], a two-stage blending algorithm was presented. In the first stage, a preliminary blended image is synthesized using the Poisson gradient loss, style loss, and content loss. In the second stage, the preliminary blended image is further transformed to have a more similar style to match the target image. It was reported that the approach works well not only for real-world target images but also stylized paintings.

Gradient domain techniques are often required to solve a large sparse linear system, therefore several fast Poisson solvers [21, 26–28] were proposed in the past. Despite the advantages of the previous Poisson solvers, the computation becomes time-consuming when applied to large images, where gradient field was obtained on a fine grid. In [29], a reduced-order finite difference method based on proper orthogonal decomposition technique was applied.

In this paper, we propose a novel fast Poisson solver that utilized red-black ordering strategy with two different relaxation factors and two accelerated parameters implemented on skewed grid. With red-black ordering, each pixel in the computed region is labeled as red or black color node. Since red and black nodes are computed independently, different relaxation factors can be applied to the two groups of nodes, thus reduces the number of iterations required during the compositing process. The additional relaxation factors and accelerated parameters provide wider choice of values to further speed up the execution time. By implementing the compositing process on a skewed grid, the total computations can be drastically reduced since only half of the pixels inside the compositing region are involved during the iteration procedure. In the previous studies, the iterative algorithms implemented on skewed grid were successfully applied to solve various problems [30–32].

The main contributions of this article are as follows:

- We propose a new image compositing method implemented on a skewed grid, in which we develop a new algorithm using red-black ordering strategy.
- We execute image compositing process from gradient field obtained by solving the Poisson equation using iterative solvers that employ additional relaxation factor and accelerated parameters for wider parameter tuning options to improve the overall computational time.
- We conduct a series of experiments, and the results demonstrate the superiority of our image compositing method.

The remainder of this paper is organized as follows. Section 2 establishes a Poisson image compositing approach that employ red-black ordering strategy on regular and skewed grids. Experimental evaluations for comparing the proposed solution on skewed grid against solutions on regular grid are presented in Section 3. Discussion on the findings of the suggested algorithms and their limitations are provided in Section 4. Finally, Section 5 presents the conclusions and an outlook on possible future work.

2. Materials and methods

2.1. Poisson image compositing

Directly copying a foreground object from source image and pasting it onto a target image can produce big intensity changes at the boundary, which creates obvious artifacts to human eyes. Therefore, the motivation of Poisson image compositing is to smooth the abrupt intensity change in the compositing boundary in order to reduce artifacts. In Figure 1, the left image shows the abrupt intensity

change in the composite boundary in the copy-and-paste image and a smooth boundary transition using Poisson compositing [8].

(a) Copy-and-Paste (b) Poisson compositing (c) Image interpolation (d) Source image

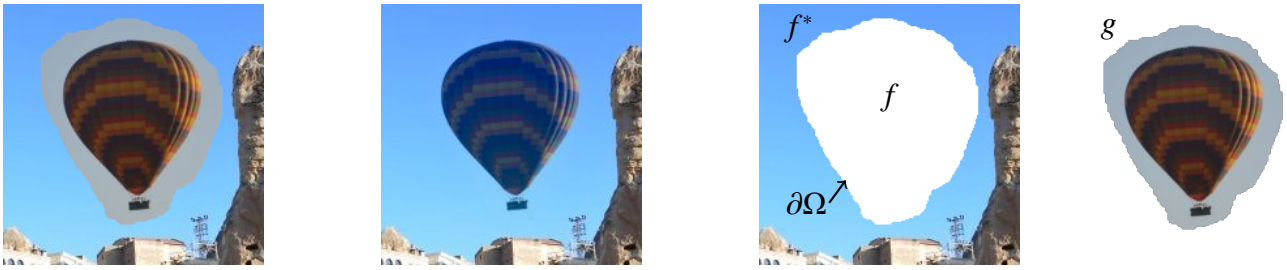

Figure 1. The two images on the left are the output of (a) "copy-and-paste" and (b) "Poisson compositing" image editing. The images (c) and (d) on the right show the formulation of image interpolation.

Based on the original work in [8], Poisson image compositing is formulated as an image interpolation problem using a guidance vector field.

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \quad \text{with} \quad f\big|_{\partial\Omega} = f^*\big|_{\partial\Omega} \tag{2.1}$$

where $\nabla = \left[\frac{\partial .}{\partial x}, \frac{\partial .}{\partial y}\right]$ is the gradient operator, f is the function of the compositing image, f^* is the function of the target image, $\mathbf{v}$ is the vector field, Ω is the compositing region and $\partial\Omega$ is the boundary of the compositing region. The solution is the unique solution of the following Poisson equation with Dirichlet boundary conditions:

$$\Delta f = \text{div}\mathbf{v} \text{ over } \Omega, \text{ with } f\,|_{\partial\Omega} = f^*|_{\partial\Omega}\,, \tag{2.2}$$

where $\text{div}\mathbf{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$ is the divergence of $\mathbf{v} = (u, v)$. This is the fundamental machinery of Poisson editing of color images: three Poisson equations of the form (2.1) are solved independently in the three color channels of the chosen color space. All the results reported in this paper were obtained in the RGB color space.

2.1.1. Iterative solver

This section describes the iterative solver to solve the Poisson equation (2.2). For discrete images, the problem can be discretized naturally using the underlying discrete pixel grid. The finite difference discretization of Eq (2.2) using the five-point Laplacian operator will result in the following approximation equation:

$$u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j} = h^2 f_{i,j}. \tag{2.3}$$

The iterative scheme of Eq (2.3) is given as follows [33]:

$$u_{i,j}^{(k+1)} = \frac{1}{4}\left[u_{i-1,j}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k+1)} + u_{i,j+1}^{(k)} - h^2 f_{i,j}\right]. \tag{2.4}$$

From Eq (2.4), the Successive Overrelaxation (SOR) [33] scheme that utilizes a relaxation factor is obtained and written as follows:

$$u_{i,j}^{(k+1)} = \frac{\omega}{4}\left[u_{i-1,j}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k+1)} + u_{i,j+1}^{(k)} - h^2 f_{i,j}\right] + (1-\omega)u_{i,j}^{(k)}. \tag{2.5}$$

In [34], the Accelerated Overrelaxation (AOR) scheme was developed and described as follows:

$$u_{i,j}^{(k+1)} = \frac{\omega}{4}\left[u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} - h^2 f_{i,j}\right] + \frac{\rho}{4}\left[u_{i-1,j}^{(k+1)} - u_{i-1,j}^{(k)} + u_{i,j-1}^{(k+1)} - u_{i,j-1}^{(k)}\right] + (1-\omega)u_{i,j}^{(k)}, \tag{2.6}$$

where an accelerated parameter ρ is added. An extension to the SOR and AOR is the Two-Parameter Overrelaxation (TOR) iterative method [35] which employs two accelerated parameters ρ and σ, to provide more tuning choices during the iteration process. The finite-difference approximation equation for the TOR method is given as follows:

$$u_{i,j}^{(k+1)} = \frac{\omega}{4}\left[u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} - h^2 f_{i,j}\right] + \frac{\rho}{4}\left[u_{i-1,j}^{(k+1)} - u_{i-1,j}^{(k)}\right] + \frac{\sigma}{4}\left[u_{i,j-1}^{(k+1)} - u_{i,j-1}^{(k)}\right] + (1-\omega)u_{i,j}^{(k)}, \tag{2.7}$$

The optimal values for the three parameters (ω, ρ and σ) are in the range between 1 and 2.

2.2. Modified iterative algorithm using red-black ordering

As shown in Figure 2, in contrast to the standard ordering in which all nodes are in one color, the modified iterative algorithm employs red-black ordering approach that sets the nodes in red and black colors [36]. By applying the red-black ordering, the computations of red and black nodes involve two phases, in which the red and black nodes are computed alternately. The procedure begins by computing red nodes where $(i + j)$ is even. In the second phase, black nodes, where $(i + j)$ is odd, are computed. As shown in Figure 3, the computations of red and black nodes $u^{(k)}$ are based on the updated values $u^{(k+1)}$ of their four neighbouring opposite color nodes. The iteration procedures of these two phases continue until the stopping criterion is satisfied.

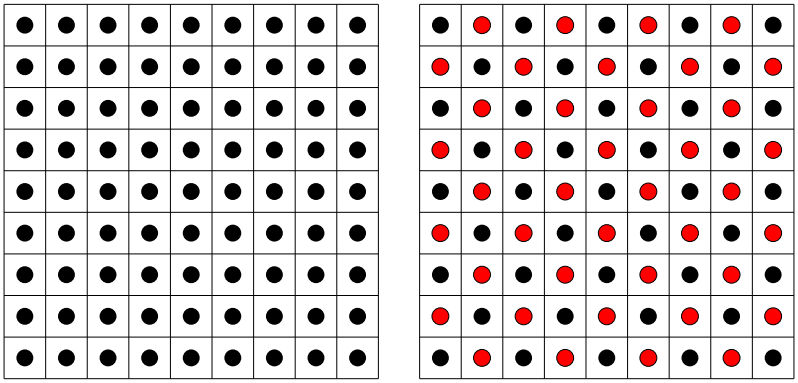

Figure 2. The regular one color (left) and two colors (right) grids.

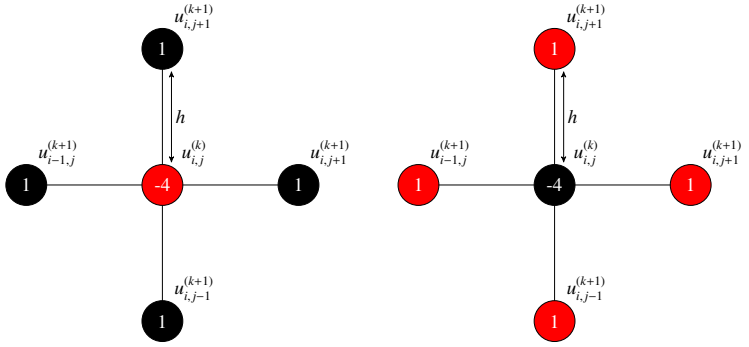

Figure 3. The computational molecules of red (left) and black (right) nodes for regular grid.

Based on the iteration scheme given in Eq (2.5), the Modified Successive Overrelaxation (MSOR) algorithm [37] can be written as

$$u_{i,j}^{(k+1)} = \frac{\alpha}{4}\left[u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} - h^2 f_{i,j}\right] + (1-\alpha)u_{i,j}^{(k)}, \tag{2.8}$$

and

$$u_{i,j}^{(k+1)} = \frac{\beta}{4}\left[u_{i-1,j}^{(k+1)} + u_{i+1,j}^{(k+1)} + u_{i,j-1}^{(k+1)} + u_{i,j+1}^{(k+1)} - h^2 f_{i,j}\right] + (1-\beta)u_{i,j}^{(k)}, \tag{2.9}$$

where α and β are the relaxation factors for red and black nodes respectively, and both values are chosen such that $0 < \alpha, \beta < 2$, where the optimal values are in the range between 1 and 2.

The Modified Accelerated Overrelaxation (MAOR) algorithm is based on Eq (2.6), where Eq (2.8) is used for the iteration of its red nodes and the iterative scheme for black nodes is written as

$$\begin{aligned} u_{i,j}^{(k+1)} = {} & \frac{\beta}{4}\left[u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} - h^2 f_{i,j}\right] + (1-\beta)u_{i,j}^{(k)} \\ & + \frac{\rho}{4}\left[u_{i-1,j}^{(k+1)} - u_{i-1,j}^{(k)} + u_{i,j-1}^{(k+1)} - u_{i,j-1}^{(k)} + u_{i+1,j}^{(k+1)} - u_{i+1,j}^{(k)} + u_{i,j+1}^{(k+1)} - u_{i,j+1}^{(k)}\right], \end{aligned} \tag{2.10}$$

where $0 < \alpha, \beta, \rho < 2$.

Based on Eq (2.7), the Modified Two-Parameter Overrelaxation (MTOR) algorithm employs two relaxation factors and two accelerated parameters. Similar to the MSOR and MAOR, the iteration for its red nodes uses Eq (2.8). Meanwhile, the iterative scheme of black nodes for the MTOR is written as

$$\begin{aligned} u_{i,j}^{(k+1)} = {} & \frac{\beta}{4}\left[u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} - h^2 f_{i,j}\right] + (1-\beta)u_{i,j}^{(k)} \\ & + \frac{\rho}{4}\left[u_{i-1,j}^{(k+1)} - u_{i-1,j}^{(k)} + u_{i,j-1}^{(k+1)} - u_{i,j-1}^{(k)}\right] + \frac{\sigma}{4}\left[u_{i+1,j}^{(k+1)} - u_{i+1,j}^{(k)} + u_{i,j+1}^{(k+1)} - u_{i,j+1}^{(k)}\right], \end{aligned} \tag{2.11}$$

where all values of relaxation factors (α and β) and accelerated parameters (ρ and σ) are between 0 and 2.

2.3. Iterative algorithm on skewed grid

Apart from standard iterative algorithm on regular grid, approximation equation for problem (2.2) can be implemented on skewed grid [38] (see Figure 4), in which the approximation equation can be

written as

$$u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1} - 4u_{i,j} = 2h^2 f_{i,j} \tag{2.12}$$

and its iterative scheme is given as

$$u_{i,j}^{(k+1)} = \frac{1}{4}\left[u_{i-1,j-1}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)} - h^2 f_{i,j}\right]. \tag{2.13}$$

Figure 5 shows the computational molecules of red and black nodes on a skewed grid.

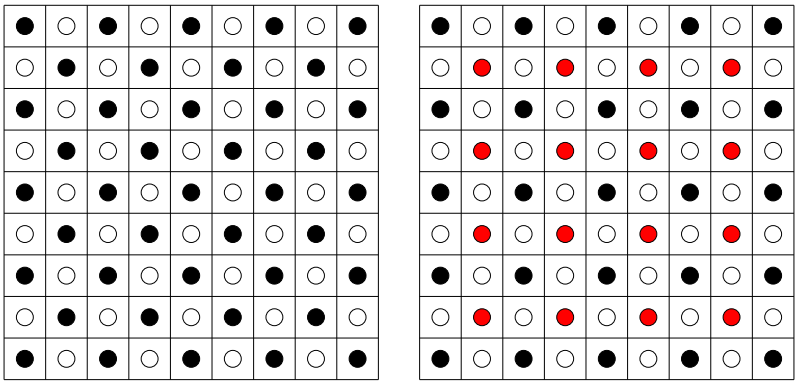

Figure 4. The standard skewed grid (left) and red-black skewed grid (right).

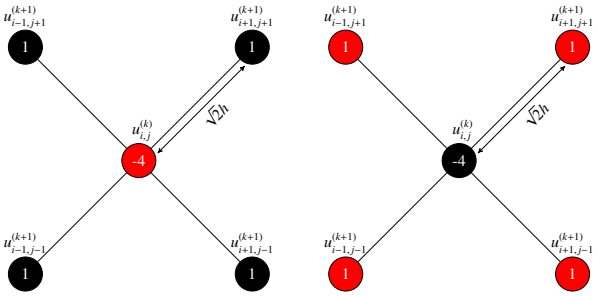

Figure 5. The skewed molecules for red (left) and black (black) nodes.

2.3.1. Skewed modified algorithms

The SOR [38], AOR, and TOR can be implemented on skewed grid using difference equation (2.12). Therefore, their corresponding iterative algorithms on skewed grid can be obtained as

$$u_{i,j}^{(k+1)} = \frac{\omega}{4}\left[u_{i-1,j-1}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)} - h^2 f_{i,j}\right] + (1-\omega)u_{i,j}^{(k)}, \tag{2.14}$$

$$\begin{aligned} u_{i,j}^{(k+1)} = {} & \frac{\omega}{4}\left[u_{i-1,j}^{(k)} + u_{i+1,j-1}^{(k)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)} - h^2 f_{i,j}\right] + (1-\omega)u_{i,j}^{(k)} \\ & + \frac{\rho}{4}\left[u_{i-1,j-1}^{(k+1)} - u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k+1)} - u_{i+1,j-1}^{(k)}\right], \end{aligned} \tag{2.15}$$

and

$$u_{i,j}^{(k+1)} = \frac{\omega}{4}\left[u_{i-1,j}^{(k)} + u_{i+1,j-1}^{(k)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)} - h^2 f_{i,j}\right] + (1-\omega)u_{i,j}^{(k)}$$

$$+ \frac{\rho}{4}\left[u_{i-1,j-1}^{(k+1)} - u_{i-1,j-1}^{(k)}\right] + \frac{\sigma}{4}\left[u_{i+1,j-1}^{(k+1)} - u_{i+1,j-1}^{(k)}\right], \tag{2.16}$$

respectively. By utilizing red-black ordering as illustrated in Figure 6, the skewed modified variants namely Skewed MSOR (SkMSOR), Skewed MAOR (SkMAOR), and Skewed MTOR (SkMTOR) can be obtained. The iterative scheme for these three algorithms on red nodes is written as

$$u_{i,j}^{(k+1)} = \frac{\alpha}{4}\left[u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)} - h^2 f_{i,j}\right] + (1-\alpha)u_{i,j}^{(k)}. \tag{2.17}$$

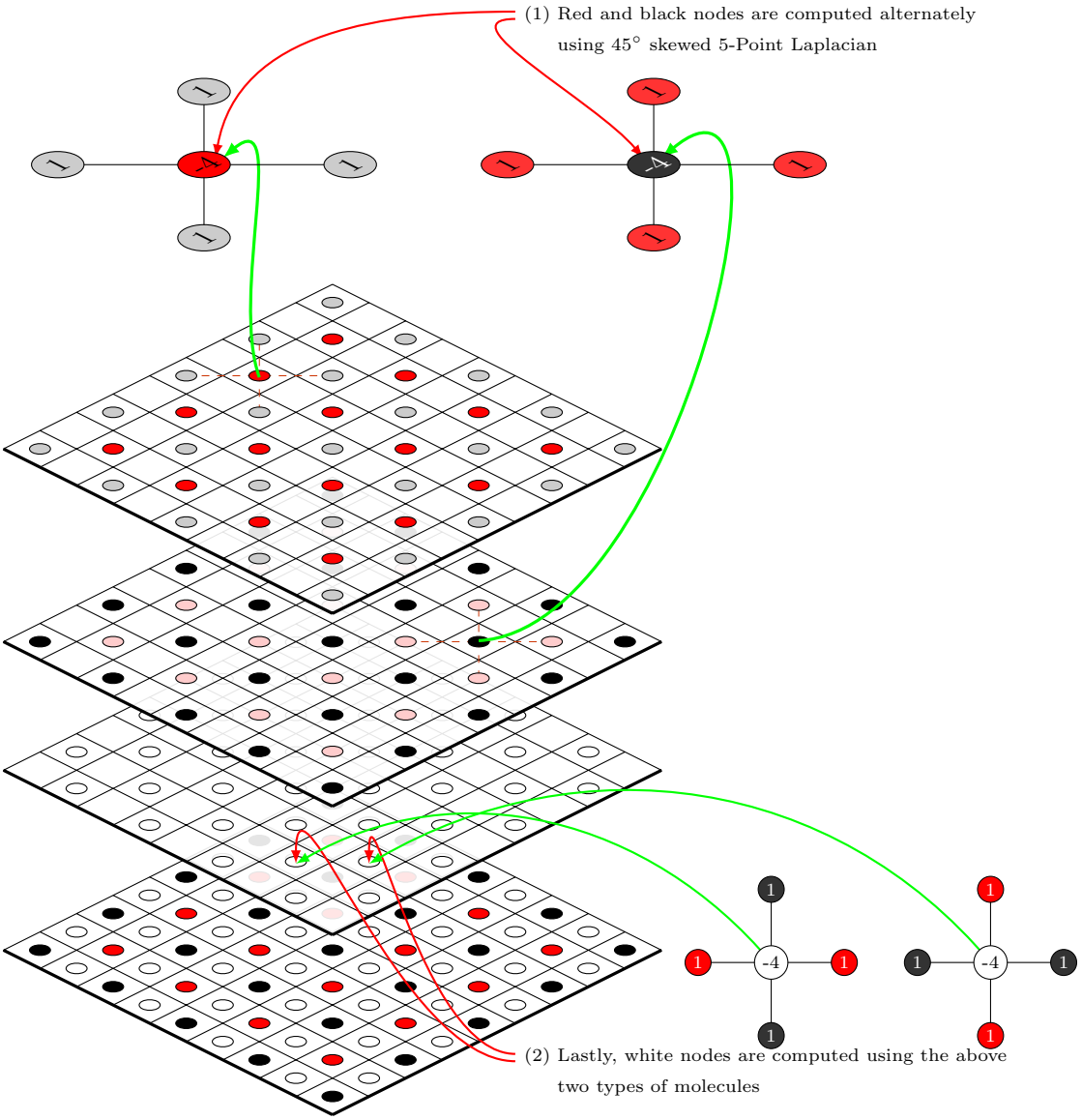

Figure 6. Illustration of red-black ordering strategy of skewed grid.

Accordingly, the iterative schemes of black nodes for the respective SkMSOR, SkMAOR and SkMTOR are given as

$$u_{i,j}^{(k+1)} = \frac{\beta}{4}\left[u_{i-1,j-1}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k+1)} + u_{i+1,j+1}^{(k+1)} - h^2 f_{i,j}\right] + (1-\beta)u_{i,j}^{(k)}, \tag{2.18}$$

$$u_{i,j}^{(k+1)} = \frac{\beta}{4}\left[u_{i-1,j}^{(k)} + u_{i+1,j-1}^{(k)} + u_{i-1,j+1}^{(k+1)} + u_{i+1,j+1}^{(k+1)} - h^2 f_{i,j}\right] + (1-\beta)u_{i,j}^{(k)}$$
$$+ \frac{\rho}{4}\left[u_{i-1,j-1}^{(k+1)} - u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k+1)} - u_{i+1,j-1}^{(k)} + u_{i-1,j+1}^{(k+1)} - u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k+1)} - u_{i+1,j+1}^{(k)}\right], \quad (2.19)$$

and

$$u_{i,j}^{(k+1)} = \frac{\beta}{4}\left[u_{i-1,j}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k+1)} + u_{i+1,j+1}^{(k+1)} - h^2 f_{i,j}\right] + (1-\beta)u_{i,j}^{(k)}$$
$$+ \frac{\rho}{4}\left[u_{i-1,j-1}^{(k+1)} - u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k+1)} - u_{i+1,j-1}^{(k)}\right] + \frac{\sigma}{4}\left[u_{i-1,j+1}^{(k+1)} - u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k+1)} - u_{i+1,j+1}^{(k)}\right]. \quad (2.20)$$

2.4. Image compositing algorithm

The algorithm is only applied to the pixels involved in the compositing process. All other pixels outside the compositing region are ignored during the computation process, thus speed up the overall execution time. By employing the red-black ordering approach, the two different relaxation factors α and β can be tuned and thus providing additional options to improve the execution time. Note that if $\alpha = \beta$, the original non-modified variants are obtained. With TOR variants, two accelerated parameters ρ and σ are available for tuning, thus provide more choice during the composition process. If $\rho = \sigma$, the AOR variants are obtained, whereas if $\rho = \sigma = 1$, we shall obtain the standard SOR variants. Algorithm 1 describes the iterative procedure for red-black ordering approach, in which the computations for red and black nodes are executed alternately until the maximum tolerance error is obtained. The `ComputePixels` function utilizes the appropriate iterative algorithm described in the previous section. For computation on skewed grid, the remaining white nodes are computed after the convergence is achieved.

Algorithm 1: Image compositing procedure for red-black ordering approach

Input: source image I_s, target image I_t, mask image I_m, tolerance error E, number of pixels N, old pixel values U, updated pixel values V
Output: output image I_o
$U \leftarrow$ InitPixels(I_s, I_t, I_m);
while $\varepsilon > E$ **do**
  for $(i, j) \in [1 : N]$ **do**
    if *IsRedNode(i, j)* **then**
      $V \leftarrow$ ComputePixels$(U, \alpha, \rho, \sigma)$
    end
  end
  for $(i, j) \in [1 : N]$ **do**
    if *IsBlackNode(i, j)* **then**
      $V \leftarrow$ ComputePixels(U, β, ρ, σ)
    end
  end
end
$I_o \leftarrow$ CopyPixels(V)

2.5. Computational complexity analysis

This section presents the analysis on the total computing costs for the considered algorithms. All algorithms were tested using the same machine running Xubuntu 18.04 on Intel i5 3470K @ 2.6GHz with 16GB memory. By assuming that the number of pixels involve in the calculation is N, the total arithmetic operations per iteration can be estimated for each algorithm. On skewed grid, only half of the total pixels, $M = \frac{N}{2}$, are involved during the computations. Meanwhile, for the modified variants that employ red-black ordering approach, the total pixels involved during the iterations of red and black nodes are M on regular grid and $m = \frac{N}{4}$ on skewed grid. We assume that the execution time for the arithmetic addition and multiplication operations require the same computing clock cycle.

For computations on a regular grid, the arithmetic operations required per iteration, for the standard algorithms (SOR, AOR, and TOR) are $7N$, $12N$, and $13N$, respectively. Their corresponding modified algorithms (MSOR, MAOR, and MTOR) require $7M$ for red nodes and $7M$, $16M$, $17M$ for black nodes, respectively. On skewed grid, the respective arithmetic operations per iteration are $7M$, $12M$, and $13M$. The skewed modified algorithms (SkMSOR, SkMAOR, and SkMTOR) require $7m$ arithmetic operations for red nodes. Their respective arithmetic operations for black nodes are $7m$, $16m$, and $17m$. The total computing cost for the considered iterative algorithms are summarized in Table 1.

Table 1. Total computing cost of the tested algorithms, where N denotes the total number of pixels, $M = \frac{N}{2}$, and $m = \frac{N}{4}$. The terms $_r$ and $_b$ indicate the number of arithmetic operations for the respective red and black nodes.

Iterative algorithms	Total arithmetic operations
MSOR	$7M_r + 7M_b = 14M$
MAOR	$7M_r + 16M_b = 23M$
MTOR	$7M_r + 17M_b = 24M$
SkMSOR	$7m_r + 7m_b = 14m$
SkMAOR	$7m_r + 16m_b = 23m$
SkMTOR	$7m_r + 17m_b = 24m$

2.6. Image quality measurements

The image quality measurement process is used to compare the similarity between the final output images. Based on the statistical method Analysis of Variance (ANOVA) [35], seven metrics are used namely Mean Square Error (MSE), Structural Similarity Index (SSIM) [39], Structural Content (SC), Normalized Cross-Correlation (NCC), Normalized Absolute Error (NAE), Average Difference (AD) and Maximum Difference (MD) [40].

The simple MSE can be written as

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(A_{ij} - B_{ij}\right)^2, \tag{2.21}$$

where A and B represent the pixel values of reference and target images, and $(m \times n)$ is the size of the image. MSE is used to measures the difference between pixel values in A and B, in which a smaller value means higher similarity. The ideal MSE value of 0 is obtained when the two images A and B

are identical. Another similarity test that can be used to compare the two images is SSIM [41]. If the obtained SSIM is equal to 1, it means the two images identical.

Similarly, SC and NCC values that are equal to 1 can be used to indicate that the two images are identical. SC value can be obtained using the following formula:

$$\mathrm{SC} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij})^2}{\sum_{i=1}^{m}\sum_{j=1}^{n}(B_{ij})^2}. \tag{2.22}$$

NCC measurement compares the two images and is given as follows:

$$\mathrm{NCC} = \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{A_{ij}\times B_{ij}}{A_{ij}^2}. \tag{2.23}$$

In contrast, NAE and AD values that are close to 0 means the two images are identical. Both values can be obtained as follows:

$$\mathrm{NAE} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(|A_{ij} - B_{ij}|)^2}{\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij})^2}, \tag{2.24}$$

and

$$\mathrm{AD} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[A_{ij} - B_{ij}], \tag{2.25}$$

where A and B are the reference and target images, respectively.

Lastly, the MD value that is used to obtain the maximum difference of pixel value between the two images A and B. MD is given the following formula:

$$\mathrm{MD} = \max(A_{ij} - B_{ij}). \tag{2.26}$$

If the obtained MD value is very small, it means the similarity between the two images is higher. All these seven metrics are used to compare the similarity of the output images generated by the six tested methods.

3. Results

In this work, three sets of images are used to measure the performance of the considered methods. The six tested methods are measured in terms of the number of iterations, execution time, and quality of images produced. All variants require optimal relaxation factor and accelerated parameter values. By conducting a trial and error procedure, initial results show that all parameter values should be in the range between 1.5 and 1.9. Based on these results, values between 1.60 and 1.74 are chosen, as shown in Table 2.

The quality of the images is compared using the similarity index as described in the previous section. Table 3 shows the number of pixels inside the image masks that were applied for the three image sets. Figure 7 shows all the image sets that comprise source, target, mask, and initial images. In Figure 8, the gradual improvement of the compositing process is illustrated. For sky and ballon images, all regular variants (MSOR, MAOR, and MTOR) take at least 600 iterations to converge. The skewed variants (SkMSOR, SkMAOR, and SkMTOR) reduce the required iterations by approximately 50%.

Table 2. Computational cost for tested images. The relaxation factors are $\alpha = 1.60, \beta = 1.66$ and accelerated parameters are $\rho = 1.70, \sigma = 1.74$. The three sets of images are: (A) Sky and ballons, (B) Grass and bird, and (C) Lake and crocodile.

		Number of iterations			CPU time (in seconds)		
Grids	Methods	A	B	C	A	B	C
Standard	MSOR	845	1211	1415	39.519	47.247	55.446
	MAOR	775	1115	1300	37.680	43.276	51.591
	MTOR	739	1065	1241	33.804	41.963	49.498
Skewed	SkMSOR	454	657	760	18.272	23.163	26.421
	SkMAOR	415	603	697	17.546	21.167	24.429
	SkMTOR	395	575	664	16.856	20.579	23.333

Table 3. Number of pixels inside the image masks.

Item	(A) Sky and ballons	(B) Grass and bird	(C) Lake and crocodile
Number of pixels	49,506	18,158	39,617

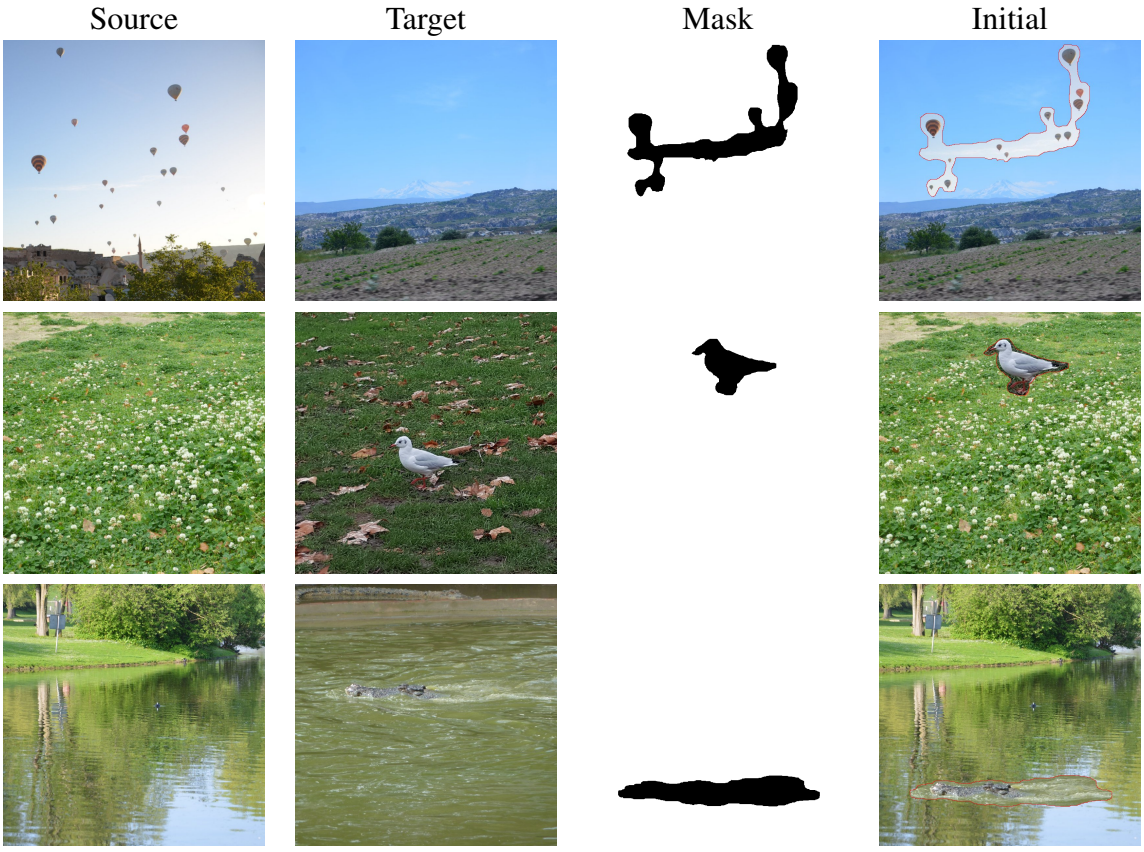

Figure 7. The source, target, mask and initial images of sky and ballons scene (top), grass and bird (middle), and, lake and crocodile (bottom).

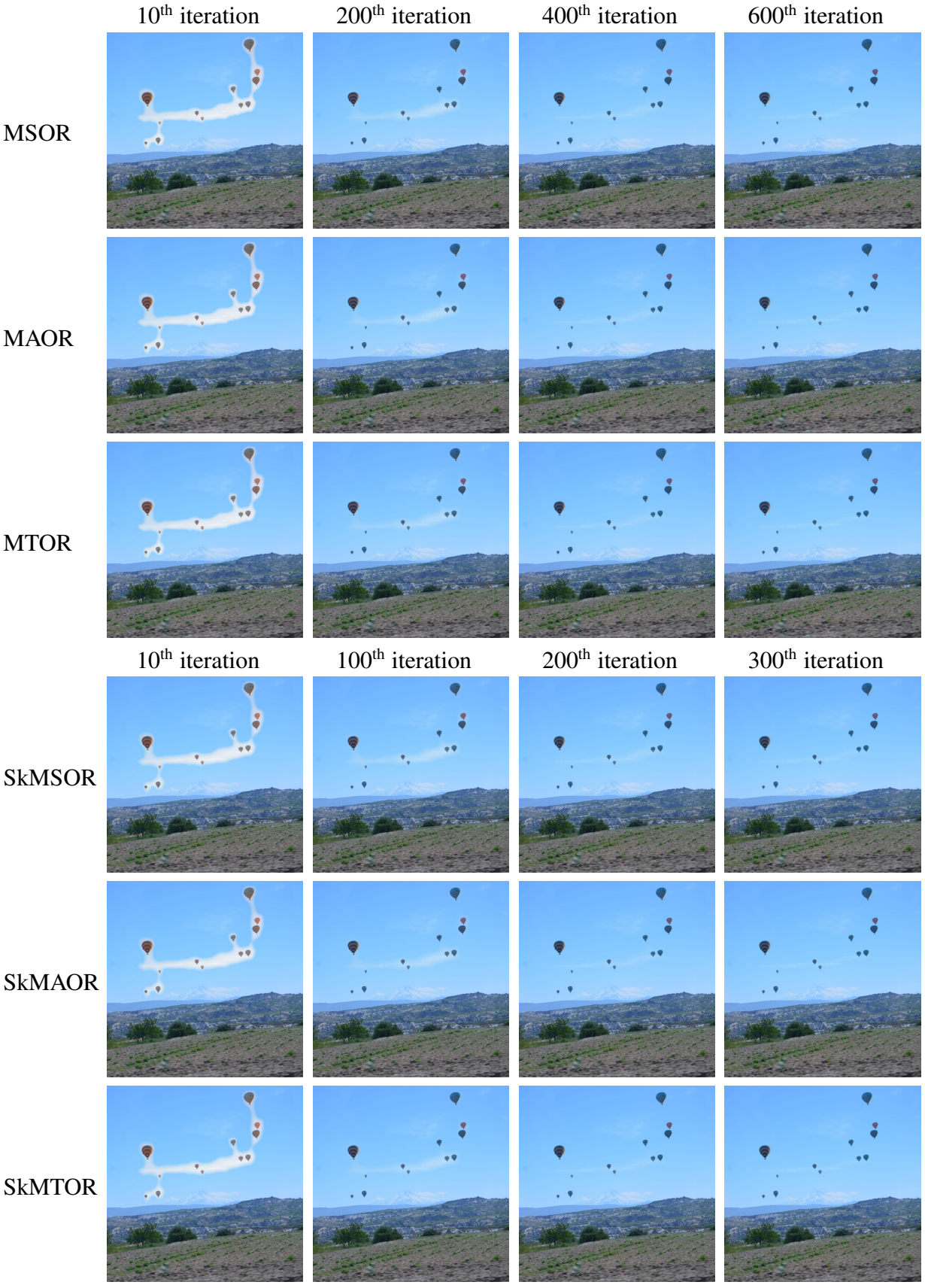

Figure 8. Illustration of compositing process at different iterations.

Table 2 shows the performance of the six tested methods. The proposed SkMTOR method gives the lowest number of iterations, thus consequently provides the best execution time. All modified skewed variants, namely SkMSOR, SkMAOR, and SkMTOR, perform faster than their corresponding regular variants of MSOR, MAOR, and MTOR. It is clearly shown from the results that the SkMTOR outperforms the other methods. Also, since all methods applied red-black ordering, the proposed methods can be implemented for parallel execution.

From Table 4, the MAOR and SkMAOR are better than the respective MSOR and SkMSOR by decreasing the iterations by 8%. The MTOR and SkMTOR slightly decrease the number of iterations by approximately 5% compared to both MAOR and SkMAOR, respectively. Both MTOR and SkMTOR are also clearly superior to the the corresponding MSOR and SkMSOR by approximately 12%. All skewed variants (SkMSOR, SkMAOR, and SkMTOR) outperforms their corresponding regular variants (MSOR, MAOR, and MTOR) by reducing the iterations by approximately 46%. In terms of CPU time, both MAOR and SkMAOR reduce the computational execution by 7% compared to the MSOR and SkMSOR, respectively. The MTOR improve the execution time by 5% compared to the MAOR, whereas the SkMTOR is slightly faster than the SkMAOR by approximately 4%. The MTOR is clearly faster than the MSOR by approximately 12%, while SkMTOR speed up the execution time by 10% compared to the SkMSOR. All skewed variants are clearly faster than their corresponding regular variants by drastically decreasing the execution time by half at approximately 51 to 52%.

Table 4. Average reduction percentage in terms of iterations and time.

Methods	Iterations reductions	Time reduction
MAOR vs MSOR	8.10%	6.796%
SkMAOR vs SkMSOR	8.34%	6.947%
MTOR vs MAOR	4.54%	5.494%
SkMTOR vs SkMAOR	4.72%	3.760%
MTOR vs MSOR	12.27%	11.917%
SkMTOR vs SkMSOR	12.67%	10.446%
SkMSOR vs MSOR	46.10%	52.285%
SkMAOR vs MAOR	46.24%	52.363%
SkMTOR vs MTOR	46.34%	51.487%

As shown in Figure 9, the generated images of grass and bird (top), and lake and crocodile (bottom) obtained from compositing process are all visually identical. No noticeable differences can be seen from all images generated by the six methods. The results of the seven metrics measurement are given in Table 5.

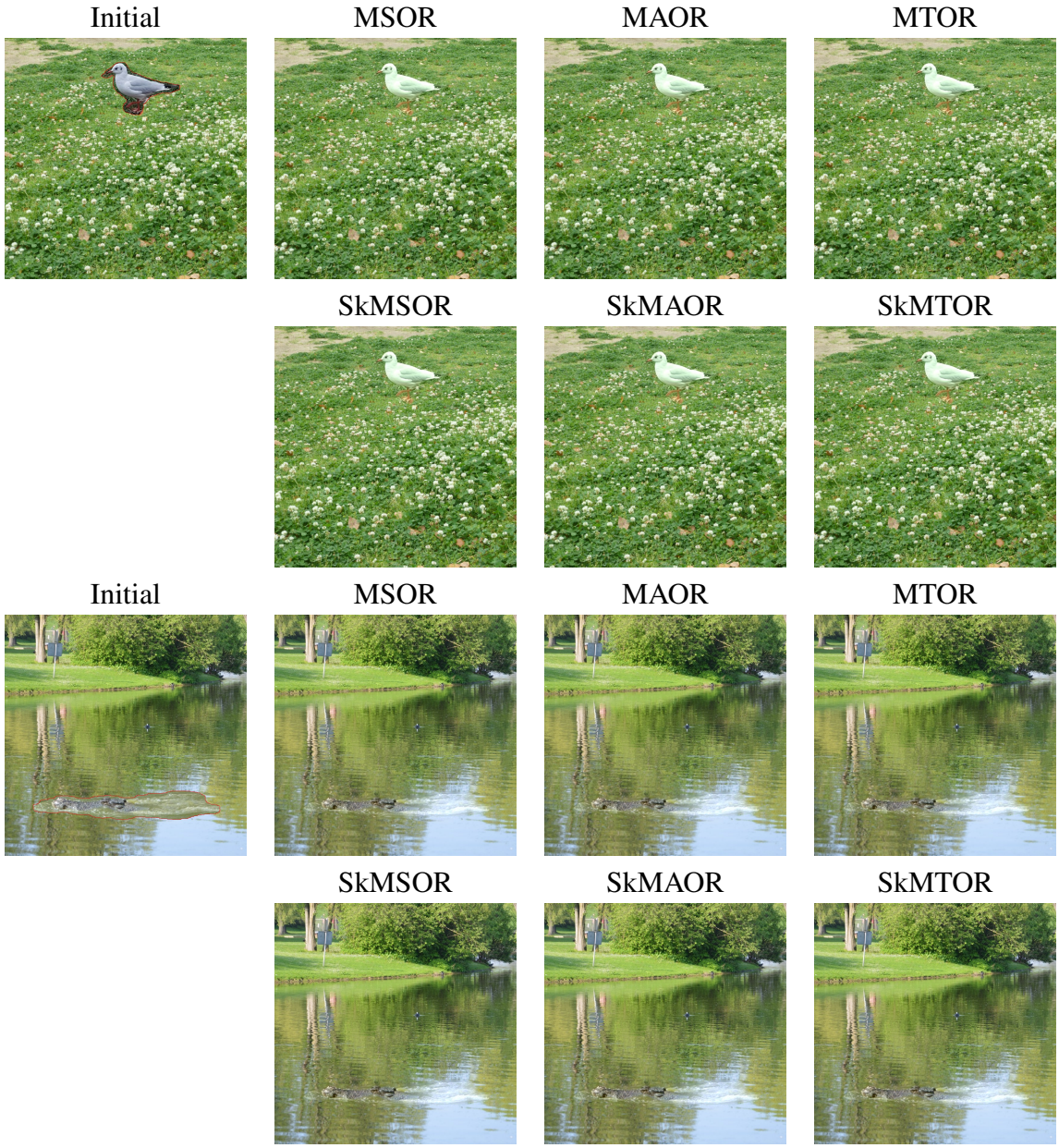

Figure 9. The output images.

Table 5. Similarity measurement for the output image.

Methods	MSE	SSIM	SC	NCC	NAE	AD	MD
MSOR	0.06678	0.99995	1.00108	0.99945	0.00005	0.05841	5
MAOR	0.07103	0.99995	1.00112	0.99944	0.00005	0.06021	5
MTOR	0.07315	0.99995	1.00113	0.99943	0.00005	0.06108	5
SkMSOR	0.07777	0.99993	1.00115	0.99942	0.00005	0.06348	5
SkMAOR	0.08120	0.99993	1.00117	0.99941	0.00005	0.06477	5
SkMTOR	0.08287	0.99993	1.00118	0.99940	0.00005	0.06538	5

4. Discussion

The findings of the algorithms suggest that the red-black technique is a viable alternative to existing approaches to tackle image composition problems. The red-black technique is well suited to parallel implementation since the computation for red and black nodes can be done independently, like on a chessboard.

Because only half of the total nodes are engaged in the iteration process, the implementation on a skewed grid minimises computational complexity. As a result, this method dramatically reduces the number of iterations and speeds up the convergence rate.

The MSE, NAE, and AD values are all close to 0 based on the obtained similarity measurement. Meanwhile, the SSIM and NCC are both close to 1, implying that the images generated are nearly identical. The highest difference (MD) between benchmark and generated image pixel intensity values is very low (less than 10). All of these similarity measurement results show that all of the algorithms examined yield identical images.

The proposed composing technique, however, requires manual fine tuning, in which optimal parameter values are obtained through a trial and error procedure. Moreover, the color of the object from the source image becomes darker or brighter, depending upon the background color of the target image.

5. Conclusions

For the six iterative approaches considered in this study, the red-black ordering implementation is employed to solve the image compositing problem. According to the findings of the experiments, the SkMTOR approach gives the fastest rate of convergence. All skewed variants outperformed their regular versions. The red-black technique, which facilitates parallelism, was not used in most prior studies. The generated images were compared using seven image similarity indexes. The image similarity tests revealed that the suggested methods produced almost identical images with no noticeable differences. More powerful point iterative approaches, such as quarter-sweep iteration, can be used to enhance the work in the future. Block-based iterative approaches would also be studied.

Acknowledgments

The authors would like to thank the Universiti Malaysia Sabah for financially supporting this research under grant SGA0117-2019.

Conflict of interest

The authors declare no conflict of interest.

References

1. S. Lin, A. Ryabtsev, S. Sengupta, B. Curless, S. Seitz, I. Kemelmacher-Shlizerman, Real-time high-resolution background matting, *arXiv:2012.07810*.

2. Z. Wang, Z. Yang, Review on image-stitching techniques, *Multimedia Systems*, **26** (2020), 413–430. doi: 10.1007/s00530-020-00651-y.
3. Y. Yuan, F. Fang, G. Zhang, Superpixel-based seamless image stitching for UAV images, *IEEE T. Geosci. Remote*, **59** (2021), 1565–1576. doi: 10.1109/TGRS.2020.2999404.
4. M. Kazhdan, M. Chuang, S. Rusinkiewicz, H. Hoppe, Poisson surface reconstruction with envelope constraints, *Comput. Graph. Forum*, **39** (2020), 173–182, doi: 10.1111/cgf.14077.
5. C. Gao, A. Saraf, J.-B. Huang, J. Kopf, Flow-edge guided video completion, In: *ECCV 2020: computer vision – ECCV 2020*, Springer, Cham, 2020, 713–729. doi: 10.1007/978-3-030-58610-2_42.
6. Z. Zhu, J. Lu, M. Wang, S. Zhang, R. R. Martin, H. Liu, et al., A comparative study of algorithms for realtime panoramic video blending, *IEEE T. Image Process.*, **27** (2018), 2952–2965. doi: 10.1109/TIP.2018.2808766.
7. O. Elharrouss, N. Almaadeed, S. Al-Maadeed, Y. Akbari, Image inpainting: A review, *Neural Process. Lett.*, **51** (2020), 2007–2028. doi: 10.1007/s11063-019-10163-0.
8. P. Pérez, M. Gangnet, A. Blake, Poisson image editing, *ACM Trans. Graph.*, **22** (2003), 313–318. doi: 10.1145/882262.882269.
9. R. Raskar, A. Ilie, J. Yu, Image fusion for context enhancement and video surrealism, In: *Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering*, 2004, 85–152. doi: 10.1145/987657.987671.
10. S. Sengupta, V. Jayaram, B. Curless, S. Seitz, I. Kemelmacher-Shlizerman, Background matting: The world is your green screen, In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, 2291–2300. doi: 10.1109/CVPR42600.2020.00236.
11. Z. Du, A. Robles-Kelly, F. Lu, Robust surface reconstruction from gradient field using the L1 Norm, In: *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)*, 2007, 203–209. doi: 10.1109/DICTA.2007.4426797.
12. D. Reddy, A. Agrawal, R. Chellappa, Enforcing integrability by error correction using L1-minimization, In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, 2350–2357. doi: 10.1109/CVPR.2009.5206603.
13. M. W. Tao, M. K. Johnson, S. Paris, Error-tolerant image compositing, *Int. J. Comput. Vis.*, **103** (2013), 178–189. doi: 10.1007/s11263-012-0579-7.
14. R. Sadek, G. Facciolo, P. Arias, V. Caselles, A variational model for gradient-based video editing, *Int. J. Comput. Vis.*, **103** (2013), 127–162. doi: 10.1007/s11263-012-0597-5.
15. J.-M. Morel, A. B. Petro, C. Sbert, Fourier implementation of Poisson image editing, *Pattern Recogn. Lett.*, **33** (2012), 342–348. doi: 10.1016/j.patrec.2011.10.010.
16. N. Limare, A. B. Petro, C. Sbert, J.-M. Morel, Retinex Poisson equation: a model for color perception, *Image Processing On Line*, **1** (2011), 39–50. doi: 10.5201/ipol.2011.lmps_rpe.
17. H. Zeng, A. Peng, X. Kang, Hiding traces of camera anonymization by Poisson blending, In: *ICAIS 2020: artificial intelligence and security*, Springer, Cham, 2020, 98–108. doi: 10.1007/978-3-030-57881-7_9.

18. K. F. Hussain, R. M. Kamel, Efficient Poisson image editing, *Electronic Letters on Computer Vision and Image Analysis*, **14** (2015), 45–57. doi: 10.5565/rev/elcvia.765.

19. M. Afifi, K. F. Hussain, MPB: A modified Poisson blending technique, *Comp. Visual Media*, **1** (2015), 331–341. doi: 10.1007/s41095-015-0027-z.

20. J. M. Di Martino, G. Facciolo, E. Meinhardt-Llopis, Poisson image editing, *Image Processing On Line*, **6** (2016), 300–325. doi: 10.5201/ipol.2016.163.

21. E. J. Hong, A. Saudi, J. Sulaiman, Numerical assessment for Poisson image blending problem using MSOR iteration via five-point Laplacian operator, *J. Phys.: Conf. Ser.*, **890** (2017), 012010. doi: 10.1088/1742-6596/890/1/012010.

22. F. Cao, T. Wu, C. Qin, Z. Qian, X. Zhang, New paradigm for self-embedding image watermarking with Poisson equation, In: *IWDW 2019: digital forensics and watermarking*, 2019, 184–191. doi: 10.1007/978-3-030-43575-2_15.

23. C. Hu, L.-Z. Huo, Z. Zhang, P. Tang, Multi-temporal landsat data automatic cloud removal using poisson blending, *IEEE Access*, **8** (2020), 46151–46161. doi: 10.1109/ACCESS.2020.2979291.

24. Y. Pan, M. Jin, S. Zhang, Y. Deng, TEC map completion using DCGAN and Poisson blending, *Space Weather*, **18** (2020), e2019SW002390. doi: 10.1029/2019SW002390.

25. L. Zhang, T. Wen, J. Shi, Deep image blending, In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, **1** (2020), 231–240. doi: 10.1109/WACV45572.2020.9093632.

26. A. Agarwala, Efficient gradient-domain compositing using Quadtrees, *ACM Trans. Graph.*, **26** (2007), 94–es. doi: 10.1145/1275808.1276495.

27. M. Kazhdan, H. Hoppe, Streaming multigrid for gradient-domain operations on large images, *ACM Trans. Graph.*, **27** (2008), 1–10. doi: 10.1145/1360612.1360620.

28. H. Zolfaghari, D. Obrist, A high-throughput hybrid task and data parallel Poisson solver for large-scale simulations of incompressible turbulent flows on distributed GPUs, *J. Comput. Phys.*, **437** (2021), 110329. doi: 10.1016/j.jcp.2021.110329.

29. H. Li, Z. Song, A reduced-order energy-stability-preserving finite difference iterative scheme based on POD for the Allen-Cahn equation, *J. Math. Anal. Appl.*, **491** (2020), 124245. doi: 10.1016/j.jmaa.2020.124245.

30. A. Saudi, J. Sulaiman, Red-black strategy for mobile robot path planning, In: *Proceedings of International MultiConference of Engineers and Computer Scientists 2010 Vol III*, 2010, 2215–2219.

31. A. Sunarto, J. Sulaiman, A. Saudi, Solving the time fractional diffusion equations by the Half-Sweep SOR iterative method, In: *2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA)*, 2014, 272–277. doi: 10.1109/ICAICTA.2014.7005953.

32. L. H. Ali, J. Sulaiman, S. R. M. Hashim, Numerical solution of fuzzy fredholm integral equations of second kind using half-sweep gauss-seidel iteration, *Journal of Engineering Science and Technology*, **15** (2020), 3303–3313.

33. D. M. Young, *Iterative solution of large linear systems*, Academic Press, 1971. doi: 10.1016/C2013-0-11733-3.

34. A. Hadjidimos, Accelerated overrelaxation method, *Math. Comput.*, **32** (1978), 149–157. doi: 10.2307/2006264.

35. J. Kuang, J. Ji, A survey of AOR and TOR methods, *J. Comput. Appl. Math.*, **24** (1988), 3–12. doi: 10.1016/0377-0427(88)90340-8.

36. D. J. Evans, Parallel S.O.R. iterative methods, *Parallel Comput.*, **1** (1984), 3–18. doi: 10.1016/S0167-8191(84)90380-6.

37. D. R. Kincaid, D. M. Young, The modified successive overrelaxation method with fixed parameters, *Math. Comput.*, **26** (1972), 705–717. doi: 10.1090/S0025-5718-1972-0331746-2.

38. A. R. Abdullah, The four point explicit decoupled group (EDG) Method: a fast poisson solver, *Int. J. Comput. Math.*, **38** (1991), 61–70. doi: 10.1080/00207169108803958.

39. Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, *IEEE T. Image Process.*, **13** (2004), 600–612. doi: 10.1109/TIP.2003.819861.

40. F. Memon, M. A. Unar, S. Memon, Image quality assessment for performance evaluation of focus measure operators, *Mehran University Research Journal of Engineering and Technology*, **34** (2015), 379–386.

41. S. H. A. Hashim, F. A. Hamid, J. J. Kiram, J. Sulaiman, The relationship investigation between factors affecting demand for broadband and the level of satisfaction among broadband customers in the South East Coast of Sabah, Malaysia, *J. Phys.: Conf. Ser.*, **890** (2017), 012149. doi: 10.1088/1742-6596/890/1/012149.

AIMS Press