An Efficient Irregular Texture Nesting Method via Hybrid NFP-SADE with Adaptive Container Resizing

Liyuan Lou, Wanyun Li, Jingle Yu, Xin Wang, and Zongqian Zhan

Abstract

Efficient irregular texture nesting, which is necessary for improving the efficiency of texture mapping and 3D model rendering, especially for large-scale 3D reconstruction tasks, has emerged as a critical research topic in the fields of photogrammetry, computer graphics, and computer vision. However, persistent inefficiencies and high computational costs in existing texture nesting algorithms pose significant challenges when dealing with vast quantities of irregularly shaped texture patches. To solve this problem, this work presents an efficient and well-structured texture nesting for reorganizing irregular textures in a space-efficient and time-efficient way. More specifically, a hybrid optimization approach that integrates an enhanced no-fit polygon (NFP) method with an improved simplified atavistic differential evolution (SADE) algorithm is proposed. The canonical SADE is reformulated, tailored for texture nesting optimization, and a novel self-adaptive container resizing strategy is used to surpass traditional NFP approaches in polygon processing efficiency. The experimental results demonstrate that the proposed method significantly improves irregular texture nesting efficiency, achieving speed improvements of up to 5.44 times compared with the common genetic algorithm–based method and 5.21 times over the simulated annealing–based method. Furthermore, it consistently improves space use by approximately 6.56%, indicating a more effective layout strategy and optimized resource use. Code is available at https://github.com/louliyuan/NFP-SADE-With-Adaptive-Container-Resizing.

Introduction

Texture nesting optimization has been a classical and fundamental research topic in the field of computer graphics and computer vision, playing a crucial role in improving 3D model rendering and texture mapping efficiency, particularly in the context of photogrammetric 3D reconstruction. In large-scale 3D reconstruction tasks, particularly those using photogrammetric techniques, texture mapping generates thousands of irregularly shaped and spatially dispersed texture patches, degrading performance and memory consumption (Weinhaus and Devarajan 1997; Ling and Qin 2023). In the absence of systematic optimization, these irregular patches lead to suboptimal loading efficiency and rendering delays (Dai *et al.* 2015).

The necessity for efficient texture nesting optimization becomes critical, as it consolidates irregularly shaped texture patches into larger and well-structured images, improving both loading efficiency and rendering performance of 3D models (Toulatzis and Fudos 2021). Despite advancements in texture mapping, existing texture nesting algorithms remain inefficient and computationally expensive, particularly in large-scale 3D reconstruction tasks. These limitations impede processing workflows and hinder the broader industrial adoption of such techniques, where both efficiency and cost-effectiveness are of paramount importance (Sheng *et al.* 2021; Ling and Qin 2023). Therefore, optimizing texture nesting efficiency is essential for reducing memory costs, enhancing productivity, and minimizing processing delays in large-scale 3D reconstruction (Junior *et al.* 2013; Liu *et al.* 2023)

To handle irregular texture patches more effectively, this paper proposes a hybrid optimization approach that integrates an enhanced no-fit polygon (NFP) method with a refined simplified atavistic differential evolution (SADE) algorithm. Moreover, a self-adaptive container resizing strategy is implemented to facilitate container grouping, thereby enhancing the efficiency of irregular polygon dataset management and optimizing both nesting efficiency and computational performance. The key contributions of this work are as follows:

1. Enhanced SADE algorithm for texture nesting optimization: The traditional SADE algorithm is refined through the incorporation of specialized mutation operations tailored to polygon placement and rotation, optimized population initialization, and dynamic fitness evaluation adjustment. Such modifications contribute to faster convergence and improved solution diversity, facilitating more efficient and high-quality nesting exploration.
2. Adaptive container grouping strategy with parallel-enabled NFP: To address the challenges in packing large-scale irregular textures, we propose an adaptive container grouping strategy that dynamically resizes and manages multiple containers during the placement process. By area-based polygon sorting and adaptive container adjustment, the method significantly improves use of space and reduces redundant computations. Furthermore, the container grouping strategy naturally facilitates parallel processing, allowing independent containers to be processed simultaneously, thus further enhancing computational efficiency. Compared with conventional NFP approaches, this method achieves a better balance between solution quality and speed.
3. Hybrid NFP-SADE framework: The enhanced NFP and SADE algorithm are integrated into a unified optimization framework. The resulting synergy leverages geometric precision of NFP for collision-free placements and the evolutionary adaptability of SADE for global optimization, thus achieving a balance between computational efficiency and nesting quality.

For clarity and systematic analysis, the remainder of this paper is structured as follows. "Related Work" reviews existing relevant texture nesting studies, followed by an overview of key concepts in "Basics." "The Proposed Hybrid Method" presents the proposed method, and "Computational Experiments" reports the experimental results. Finally, "Conclusion" offers our conclusion and outlook for future works.

Liyuan Lou, Wanyun Li, Jingle Yu, Xin Wang, and Zongqian Zhan are with the School of Geodesy and Geomatics, Wuhan University, Wuhan, Hubei Province, 430072, China.

Corresponding author: Xin Wang (xwang@sgg.whu.edu.cn)

Received March 22, 2025, accepted June 16, 2025.

Photogrammetric Engineering & Remote Sensing
Vol. 91, No. 11, November 2025, pp. 1–XX.
0099-1112/22/1–XX
© 2025 American Society for Photogrammetry and Remote Sensing
doi: 10.14358/PERS.25-00038R3

Related Work

In this section, three relevant topics are reviewed: First, classical bin packing strategies and the NFP framework, both of which provide foundational algorithms for spatial layout optimization and can be adopted in texture nesting, are studied. Second, the heuristic algorithms specifically designed for nesting optimization tasks are discussed. Finally, some relevant deep learning–based methods are reviewed.

Classical Nesting Algorithms and NFP-Based Techniques

Classical Bin Packing Approaches

Classical bin packing algorithms have long served as foundational tools for spatial layout optimization, including texture nesting applications. These methods, originally developed for rectangular or regularly shaped objects, aim to maximize use of space within constrained containers and remain relevant for polygonal texture layout tasks.

Among these, guillotine-based methods have attracted considerable attention because of their simplicity and recursive structure. The two-dimensional cutting and packing problem was first introduced by Gilmore and Gomory (1965) and has since been extensively studied. Early research focused on rectangular items with orthogonal cuts, where bins were iteratively divided using horizontal or vertical partitions (Mumford-Valenzuela *et al.* 2004; Charalambous and Fleszar 2011; Zhang *et al.* 2016). While efficient for regular shapes, these methods exhibit limitations when applied to irregular geometries. Recent advancements have extended guillotine strategies to handle irregular convex polygons by permitting nonorthogonal cuts, thereby reducing material waste through dynamic edge matching and separation cuts (Bennell *et al.* 2018). Notably, Bennell *et al.* (2018) proposed a beam search heuristic that integrates best-match edge alignment with adaptive separation strategies, achieving use rates of 85%–90% across single- and multi-bin scenarios. Their approach significantly outperformed prior metaheuristics (Martinez-Sykora *et al.* 2015) in computational efficiency, reducing run time by up to 80% for large-scale instances while maintaining solution quality through a hybrid evaluation framework.

Skyline-based algorithms have similarly evolved to include increasingly sophisticated placement strategies. Wei *et al.* (2011), for instance, introduced a rectilinear skyline model that dynamically tracks available space using adaptive skyline segments. This method emphasizes edge-aligned placements to minimize spatial fragmentation and uses a multi-criteria evaluation strategy incorporating spread constraints and fitness-based rules. *A* notable innovation is the integration of tabu search, which promotes solution diversity through strategic rectangle swaps. This hybrid approach enhances packing density and mitigates premature convergence by effectively escaping local optima. As a result, this method demonstrates improved robustness and adaptability over traditional skyline heuristics, especially in complex layout scenarios.

Collectively, these classical bin-packing strategies, especially guillotine- and skyline-based approaches, constitute the algorithmic foundation of modern spatial layout optimization. Their continuous refinement through heuristic improvements, hybridization, and geometric generalization has expanded their applicability to irregular and large-scale packing tasks. Although originally tailored for regular geometries, the core principles—recursive subdivision, skyline profiling, and heuristic-based placement—remain essential in contemporary texture nesting workflows. When integrated with broader optimization frameworks and domain-specific constraints, these strategies offer interpretable and high-performance solutions for both academic and industrial applications.

NFP in Irregular Packing

The NFP plays a foundational role in addressing irregular packing problems, particularly in enabling collision-free placement of complex shapes. Early theoretical contributions by Stoyan and Ponomarenko (1977) introduced the geometric basis of NFP through Minkowski sums, allowing precise boundary computation for convex objects. Dean *et al.* (2006) subsequently enhanced computational efficiency by proposing a vector-based approach, streamlining overlap detection in irregular nesting tasks.

Recent developments have expanded the applicability of NFP to handle nonconvex polygons and dynamic datasets. Jiang *et al.* (2016) proposed a two-stage NFP algorithm that combines convex decomposition with sliding vector operations, achieving a 15% improvement in packing density compared with traditional methods. Similarly, Xu *et al.* (2017) integrated NFP with hybrid heuristics—such as simulated annealing (SA)—to demonstrate its versatility in large-scale industrial cutting applications. Aji *et al.* (2015) further applied NFP in distributed systems for spatial partitioning, highlighting its scalability in parallel processing environments.

Despite these advances, traditional NFP techniques still encounter challenges in computational efficiency and use of space, particularly when processing large-scale or highly diverse texture datasets. Many existing implementations rely on sequential container allocation strategies (Junior *et al.* 2013), leading to redundant overlap checks and limited parallelism. These limitations are directly addressed by our enhanced NFP approach, which incorporates adaptive container grouping to improve computational efficiency and layout performance.

Heuristic Algorithms for Nesting Optimization

Heuristic algorithms have seen significant advancements in recent years, particularly through hybrid approaches that combine genetic algorithms (GAs) with other optimization techniques such as SA-based and particle swarm optimization to tackle complex texture nesting problems. For instance, Xu and Zhou (2023) proposed a hybrid algorithm that combines improved GAs with tabu search to address two-dimensional rectangular nesting problems. This method uses adaptive selection, crossover, and mutation operators, along with local search strategies during the convergence phase, to improve material use. Kang *et al.* (2024) introduced an enhanced GA integrated with a fusion tabu search strategy to solve three-dimensional bin packing problems (3D-BPPs). Their approach uses a wall-building method to generate packing solutions under residual space constraints, incorporating adaptive fitness variation and chromosome adjustment strategies to enhance population diversity and convergence speed. Furthermore, tabu search optimizes the balance between global and local search capabilities, allowing the algorithm to escape local optima and enhance packing efficiency.

Additionally, Kang *et al.* (2012) proposed a hybrid GA for the 3D-BPP, using an improved depth-bottom-left-fill strategy. This method dynamically manages packing space objects and optimizes the packing sequence through an adaptive crossover and mutation mechanism. Sun *et al.* (2025) introduced a hybrid chaotic evolutionary particle swarm optimization algorithm for two-dimensional packing problems involving conflict and load-balancing constraints. Their approach enhances local search capabilities via chaotic logistic mapping and incorporates elite crossover and dynamic mutation to avoid premature convergence. These two methods advance heuristic algorithms in complex nesting problems by integrating spatial partitioning strategies and hybrid optimization mechanisms, offering valuable insights into handling geometric constraints and improving computational efficiency in texture nesting optimization.

Deep Learning–Based Methods

Relevant deep learning–based studies have been widely applied to various scene representations, including volumes, point clouds, meshes, and implicit functions (Liu *et al.* 2015; Achlioptas *et al.* 2018; Huang *et al.* 2018; Kanazawa *et al.* 2018; Wang *et al.* 2018; Chen and Zhang 2019; Mescheder *et al.* 2019; Sitzmann *et al.* 2019; Niemeyer *et al.* 2020). Hertz *et al.* (2020) proposed a generative adversarial network (GAN)–based framework for mesh-aware geometric texture synthesis, enabling parameterization-free texture transfer across arbitrary genus shapes through vertex displacement in normal and tangential directions. This approach overcomes the limitations inherent to conventional 2D displacement maps.

Based on GANs, the spatially adaptive normalization model proposed by Park *et al.* (2019) is particularly well suited for complex scene stitching because it dynamically adjusts generation parameters for different regions through spatially adaptive normalization layers, effectively preserving the detailed information of the input semantic layout. Additionally, convolutional neural networks are used to synthesize textures for large-scale environments. For instance, Dumoulin *et al.* (2016) proposed a framework for texture synthesis with generative networks that can learn to map the high-level attributes of textures to pixel-level details. Moreover, deep learning techniques for optimizing texture nesting have become more sophisticated, with models trained on large-scale datasets to handle specific complex texture synthesis and mapping tasks (Gatys *et al.* 2015; Bergmann *et al.* 2017; Xian *et al.* 2018; Xu *et al.* 2021; Efros and Freeman 2023; Lin *et al.* 2023; Fan *et al.* 2024).

Graph-based deep learning methods have also introduced alternative solutions to texture mapping and nesting problems, emphasizing a balance between global consistency and local detail optimization. Teimury *et al.* (2020) introduced Graph-Seam, a supervised graph-based learning framework that automates UV mapping via graph neural networks to predict seam placements, thus reducing distortion and seam length while replicating artist-desired seam styles Similarly, Rouhani *et al.* (2020) proposed a noniterative global texture alignment method that selects optimal keyframes for each mesh face and uses a geometry-aware matching technique for seamless texture reconstruction, resulting in efficient large-scale texture mapping.

Further extending graph-based deep learning, Dharma *et al.* (2022) developed a graph-GAN for 3D texture generation, using unsupervised learning to extract object component information. This approach eliminates the need for costly forward passes when altering camera viewpoints or lighting, enabling generalization to unseen 3D meshes and further improving the practicality of graph-based approaches. Notably, the graph convolutional networks proposed by Kipf and Welling (2016) further enhance graph-based learning by efficiently propagating node features through localized spectral filters, achieving scalable semisupervised classification with linear computational complexity in the number of graph edges. Their framework avoids explicit graph regularization and adaptively learns representations from both labeled and unlabeled data, offering potential optimizations for modeling texture-to-mesh relationships.

Despite recent advances, many deep learning–based approaches remain constrained by task specificity, high data requirements, and substantial computational overhead, which limits their direct applicability to geometry-driven texture nesting tasks. Future research may focus on developing lightweight architectures and hybrid frameworks to overcome these challenges.

Basics

To enhance clarity in presenting our proposed method, this section first introduces two fundamental basics that underpin its operation: the NFP and the SADE algorithm.

The NFP

The NFP is a geometric computational tool used to determine the potential contact positions of polygons. It has proven to be an effective method for solving a variety of problems, such as two-dimensional irregular cutting-stock problems (Junior *et al.* 2013). The NFP is widely applied across various fields, including computer graphics, image processing, and geometric optimization (Dean *et al.* 2006; Jiang *et al.* 2016; Xu *et al.* 2017; Wang, Gu *et al.* 2024; Wang, Zhang *et al.* 2024).

In general, the NFP is defined as follows: given two polygons, A (the fixed piece) and B (the orbital piece), and a reference point on B, denoted R_B, the NFP is a geometric shape that defines the set of all potential positions of B relative to A, where B remains in contact with A but does not overlap. The trajectory traced by reference point R_B, as it moves along the contour of A, forms a closed polygon that represents the NFP of A relative to B (Wang, Zhang *et al.* 2024), as shown in Figure 1.

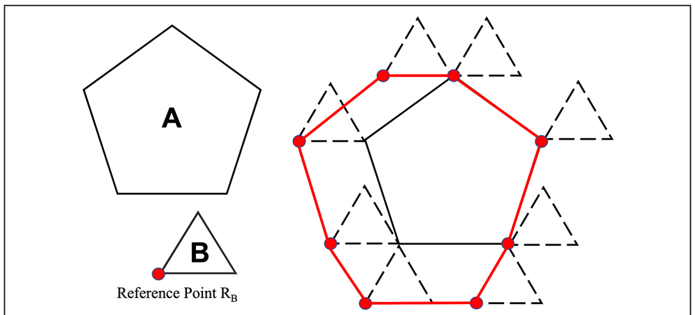


Figure 1. No-fit polygon generation process.

In our implementation, since the nesting objects in this study are irregular polygons, we use the Minkowski sum (Ghosh 1993), a concept involving the combination of two arbitrary point sets, A and B, to calculate the NFP between convex polygons. The Minkowski sum is the result of adding each point in set A to every point in set B, which is expressed as:

$$A \oplus B = \{a + b : a \in A,\ b \in B\} \tag{1}$$

The union of the geometric elements of a set can also be defined by the Minkowski vector sum. If Ab represents the set A translated by vector B, then

$$S = A \oplus B = \bigcup_{b \in B} A_b \tag{2}$$

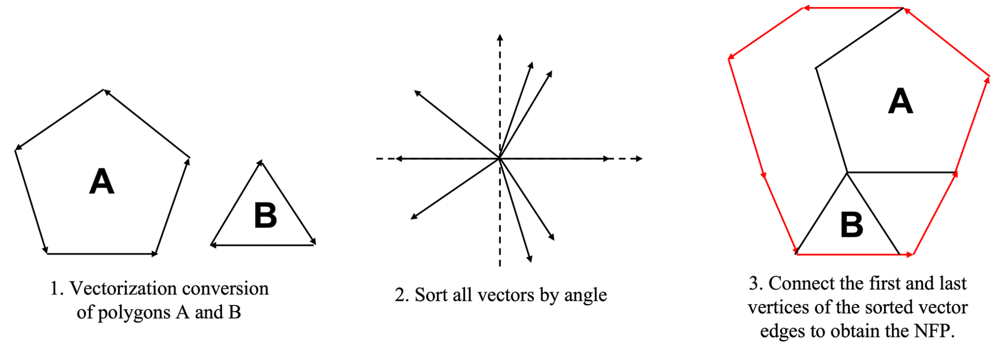
The above definition assumes that both polygons share the same orientation, in which case the NFP is not produced. However, if polygon B is transposed to its symmetric set

$$B' = \{-b : b \in B\} \tag{3}$$

then A and B' would share the same orientation, enabling the generation of the NFP. Stoyan and Ponomarenko (1977) initially established the relationship between the Minkowski vector method and normals, with a more rigorous proof provided in their work. They refer to the NFP as the hodograph. Therefore, the process of calculating the critical polygon for convex polygons A and B is as follows:

1. First, vectorize polygon A in the counterclockwise direction and polygon B in the clockwise direction.
2. Place the starting points of all vector edges of polygon A at the origin (0, 0) and the same for polygon B.
3. Sort all vector edges of polygons A and B in ascending order according to the angle between each vector and the reference vector.
4. Finally, connect the sorted vector edges sequentially from head to tail to obtain the critical polygon for A and B.

The computational diagram for convex polygons A and B is shown in Figure 2.


A
B
1. Vectorization conversion of polygons A and B
2. Sort all vectors by angle
A
B
3. Connect the first and last vertices of the sorted vector edges to obtain the NFP.
Figure 2. Simple example of Minkowski vector and normal method for generating no-fit polygon (NFP).

SADE Algorithm

The SADE algorithm was developed as an extension of traditional differential evolution (DE) via extensive research and iterative developments (Hrstka and Kučerová 2000). It is designed to address complex optimization problems in continuous domains, particularly for those with a large number of variables. SADE integrates key features of DE with elements of GAs, combining the simplified differential operator of DE with a framework similar to that of GAs. This hybridization improves its adaptability and efficiency, making it particularly suitable for large-scale optimization tasks (Hrstka and Kučerová 2004).

A comprehensive description of the SADE method, including the algorithm scheme, operators, and documentation for tests on high-dimensional optimization problems, can be found in Hrstka and Kučerová (2000).

The SADE algorithm tends to form clusters of candidate solutions, which dynamically evolve across the search space. This behavior shares similarities with gradient-based optimization methods, but differs in several key aspects. First, SADE evaluates multiple candidate solutions simultaneously, allowing it to more effectively identify subregions that contain optimal solutions. Furthermore, since the evolution of individuals is influenced by their relative distances, the algorithm dynamically adjusts step sizes, which enhances its ability to efficiently converge toward an optimal solution. More details regarding the implementation of the SADE algorithm may be seen in Table 1.

Table 1. Algorithm 3.1: simplified atavistic differential evolution (SADE).

Algorithm 3.1 Simplified Atavistic Differential Evolution (SADE)

```
1: Initialize population $P$
2: Initialize crossover rate and mutation strategy selection rate
3: while the termination condition is not met do
4:    for individual $i$ in $P$ do
5:       Select a mutation strategy
6:       Generate the mutation vector $V$
7:       Perform the crossover operation to generate the candidate solution $U$
8:       if fitness($U$) > fitness($P[i]$) then
9:          $P[i] = U$
10:      end if
11:   end for
12:   Adaptively update the crossover rate and mutation strategy selection rate
13: end while
```

The Proposed Hybrid Method

In this section, we report a detailed overview of the overall workflow of our new hybrid method focusing on two main contributions: 1) enhanced NFP method and 2) improved SADE algorithm.

Overview of Our Hybrid Nesting Method

Figure 3 shows the general workflow and main components of our proposed method, which integrates two primary parts: NFP generation and the SADE process. The pseudocode of the proposed hybrid method is provided in Table 2.

To enhance clarity, the definitions of several critical parameters are provided. The population size P denotes the number of individuals preserved in each generation throughout the evolutionary process. This parameter was empirically calibrated through preliminary experimentation to achieve a trade-off between solution accuracy and computational efficiency. The termination condition in the algorithm is defined by two factors: the number of iterations and the optimal fitness value. Specifically, the algorithm terminates when the number of iterations reaches the predefined maximum ($Iterations < MaxIter$) or when the optimal fitness value of the best individual falls below a predetermined threshold ($f_{\text{best}} < T$). The target fitness value T corresponds to a predetermined threshold that reflects an acceptable level of solution quality. Its value was established in accordance with practical application requirements and informed by empirical observations derived from previous studies.

Table 2. The proposed hybrid texture nesting method.

Algorithm 4.1 Hybrid Optimization Method

```
Input: Polygon dataset, population size $P$, mutation rate $MR$, termination condition
Output: Optimized texture layout solution
1: Sort polygons in descending order of area
2: Estimate initial container size using total polygon area
3: Initialize primary and secondary containers
4: Generate initial population $P$ with feasible placements and rotations
5: while termination condition not met do
6:    Calculate and obtain all the NFP between polygons
7:    for each polygon in sorted dataset do
8:       Attempt placement in primary container using NFP
9:       if placement fails then assign to secondary container
10:   end for
11:   Apply adaptive resizing on primary container if fill ratio < $SUR$
12:   Evaluate fitness for each individual in the population
13:   Sort $P$ by fitness in ascending order
14:   Preserve best individual
15:   for each selected individual $CH_i$ do
16:      Generate random reference individual $RP$
17:      Compute mutation vector:
18:   $CH_k(t+1) = CH_i(t) + MR \times (RP - CH_i(t))$
19:      Ensure feasibility of mutated individual
20:   end for
21:   Combine original and mutated population
22:   while population size $> P$ do
23:      Randomly select two individuals
24:      Discard the one with worse fitness
25:   end while
26: end while
```

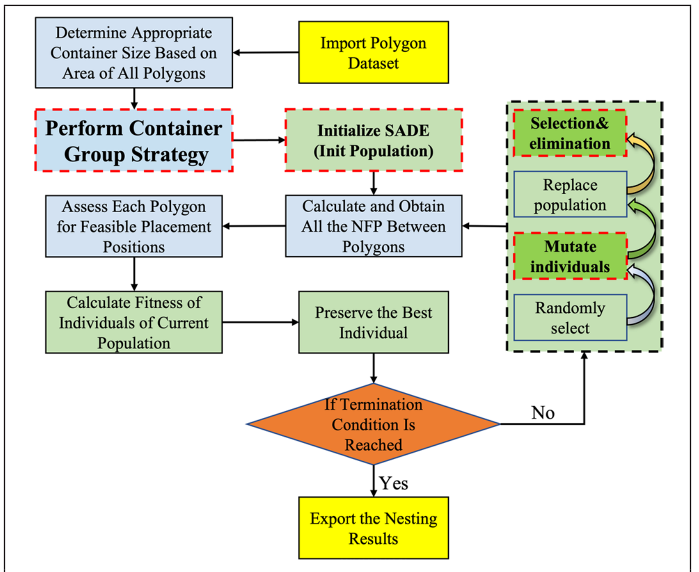

Determine Appropriate Container Size Based on Area of All Polygons
Import Polygon Dataset
Perform Container Group Strategy
Initialize SADE (Init Population)
Selection& elimination
Replace population
Mutate individuals
Randomly select
Assess Each Polygon for Feasible Placement Positions
Calculate and Obtain All the NFP Between Polygons
Calculate Fitness of Individuals of Current Population
Preserve the Best Individual
If Termination Condition Is Reached
No
Yes
Export the Nesting Results

Figure 3. The overall workflow of the proposed method. NFP = no-fit polygon; SADE = simplified atavistic differential evolution.

NFP Generation

First, the polygon dataset is sorted by area size in descending order, and an appropriate container size is then determined based on the total area of all polygons. Second, unlike the traditional NFP method, our approach incorporates the container group strategy with parallel computation, where placed polygons are assigned to a primary container, while unplaced ones are directed to a secondary container for concurrent processing. Finally, all NFPs between polygon pairs are computed, and each polygon is assessed for feasible placement positions. More details can be found in "Enhanced NFP Method."

SADE-Based Optimization

To efficiently solve the texture nesting problem, we adopt an improved SADE algorithm tailored for geometric layout optimization. The SADE component operates on a population-based evolutionary framework, starting with the initialization of a diverse and constraint-compliant population. Each individual encodes the placement and rotation of all polygons and is evaluated using the fitness function. The optimization proceeds iteratively via a main evolutionary loop, where individuals undergo mutation based on a randomized differential strategy, followed by fitness evaluation and selection through a modified tournament mechanism. Notably, individuals violating geometric constraints are discarded or penalized during evaluation, ensuring the feasibility of generated layouts. The algorithm terminates when either a maximum number of generations is reached or a convergence threshold is met. This design leverages the global search capability of SADE while integrating geometric feasibility checks from the NFP module to improve convergence speed and solution quality. (See more details in "Improved SADE.")

Enhanced NFP Method

As the number of polygons increases, the computational cost and time required for NFP calculations grow significantly. The traditional container allocation method exacerbates these challenges by demonstrating low space use and excessive resource waste, which become more pronounced with complex polygon shapes or uneven dataset distributions, leading to decreased computational efficiency and higher operational costs (Aji *et al.* 2015; Zuo *et al.* 2022). The workflow of the enhanced NFP process is illustrated in Figure 4.

To address these limitations, this paper introduces two key improvements to the traditional NFP approach, with a particular emphasis on a container group strategy designed to enhance scalability and adaptability in large-scale texture layout tasks.

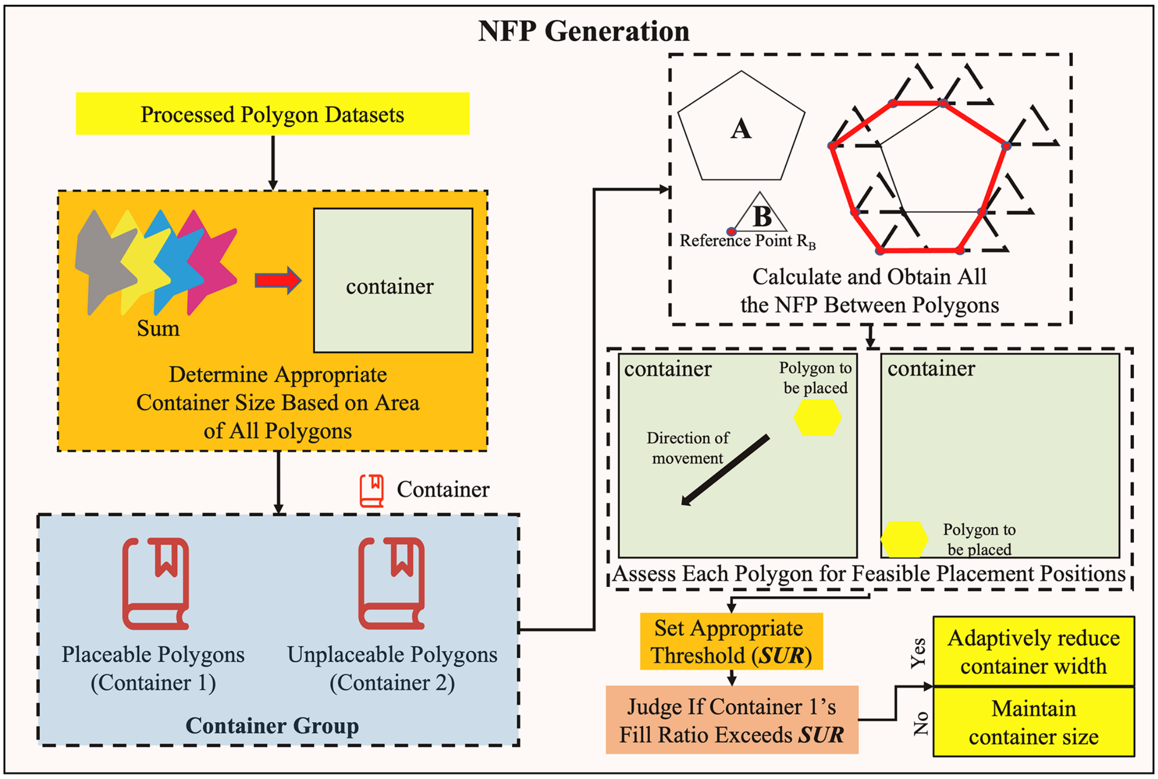
Specifically, the process begins by estimating an initial container size based on the total area of all polygons. This estimation is rounded to the nearest power of two to ensure compatibility with GPU-based texture rendering and mipmapping. If the fill ratio of the container falls below a predefined threshold (*SUR [Spatial Utilization Ratio]*), its width is dynamically reduced to improve packing compactness while maintaining placement feasibility.

Subsequently, a container group strategy is applied to manage the placement process. All polygons are initially sorted in descending order of area, and then processed in parallel. Polygons that can be feasibly placed are assigned to the primary container, while those that cannot be placed because of geometric constraints are redirected to a secondary container for concurrent processing. This design improves computational efficiency by isolating unplaceable or complex shapes into a separate container, thereby reducing placement conflicts and redundant overlap computations. Each container manages its own placement logic independently, which is particularly effective when handling texture datasets with a high degree of shape diversity or uneven polygon distribution. While only two containers are used, this mechanism achieves partial parallelism and improves throughput by localizing computational complexity.

Together, the adaptive resizing mechanism and dual-container strategy enable the proposed approach to efficiently handle diverse and large-scale polygon datasets. By localizing complexity and dynamically adjusting space allocation, the method achieves a favorable balance between computational efficiency and packing quality, making it particularly suitable for real-world texture layout applications with high performance demands.

Improved SADE

To address the inefficiencies of existing texture nesting algorithms, this work incorporates various optimization strategies into the classical



NFP Generation
Processed Polygon Datasets
Sum
container
Determine Appropriate Container Size Based on Area of All Polygons
Container
Placeable Polygons (Container 1)
Unplaceable Polygons (Container 2)
Container Group
A
B
Reference Point R_B
Calculate and Obtain All the NFP Between Polygons
container
Polygon to be placed
Direction of movement
container
Polygon to be placed
Assess Each Polygon for Feasible Placement Positions
Set Appropriate Threshold (*SUR*)
Judge If Container 1's Fill Ratio Exceeds *SUR*
Yes
No
Adaptively reduce container width
Maintain container size

Figure 4. Enhanced no-fit polygon (NFP) processing workflow.

SADE algorithm, yielding a more effective texture nesting optimization algorithm based on SADE. Figure 5 presents the workflow of our improved SADE process and the key modules. Specifically, the process begins with the preparation of a polygon dataset and the enhanced NFP generation. If the SADE algorithm has not yet been initialized, the process proceeds with population initialization; otherwise, the population is sorted in ascending order according to fitness, and the best individual is preserved. Then, the main loop is executed.

Initialize SADE (Init Population)

The initialization phase of the SADE algorithm plays a crucial role in establishing a diverse and feasible starting point for the evolutionary process, which directly affects convergence behavior and overall performance. In this phase, each individual in the population is randomly generated within the feasible solution space to ensure diversity and broad coverage of the search domain (Hrstka and Kučerová 2004). Parameters such as placement positions and rotation angles are typically sampled from a uniform distribution to avoid any initial bias toward specific regions. Importantly, all generated individuals are required to satisfy problem-specific constraints from the outset, guaranteeing that the initial population is valid and suitable for evolution. *A* fixed population size P is maintained, with each individual represented as a vector of decision variables tailored to the structure of the problem. This well-initialized population forms the foundation for subsequent evolutionary operations, such as mutation and selection, and provides the algorithm with a robust starting point for exploring the solution space effectively.

Main Loop of the Improved SADE

The main loop of the improved SADE algorithm iteratively executes a sequence of operations until a termination condition is met, which occurs either when the maximum number of iterations is reached or when the optimal fitness value falls below a predefined threshold. At each generation, individuals are randomly selected, undergo mutation operations, and are subsequently evaluated. The population is then updated through a selection and elimination process that retains the most promising solutions.

Mutation plays a pivotal role in maintaining diversity and promoting global exploration in the evolutionary process. In this framework, mutation is primarily implemented through the *Mutation* operator, which introduces stochastic perturbations into selected individuals while preserving feasibility. Specifically, when an individual binary string (so-called chromosome) $CH_i(t)$ is selected for mutation, a new random chromosome RP is generated, and the updated chromosome $CH_k(t+1)$ is computed using the following strategy (Hrstka et al 2003):

$$CH_k(t+1) = CH_i(t) + MR\,(RP - CH_i(t)) \tag{4}$$

Here, MR is the mutation rate, controlling the scale of the variation introduced. This formulation ensures that new individuals are generated in a direction guided by a randomly sampled reference, thereby balancing exploration and exploitation. All mutations are constrained within the valid solution space, ensuring that geometric feasibility and placement constraints are respected.

Following mutation, the algorithm proceeds with a selection and elimination phase to refine the population. Inspired by a modified tournament selection strategy, two individuals are randomly selected from the combined population (original and mutated), and the one with the worse fitness is discarded. This elimination step is repeated iteratively until the population size is reduced back to the predefined constant P. This elitist yet diversity-preserving approach ensures that high-quality individuals are retained while maintaining sufficient variability to avoid premature convergence.

Time Complexity Analysis

The time complexity of our method is discussed as well. Given that the population size is defined as N, the maximum number of iterations as $MaxIter$, and the complexity of the *find_fitness* function as $O(f)$, we evaluate the computational cost accordingly. First, the complexity of generating the population is $O(N)$, where the fitness values of N individuals in each population are calculated sequentially, resulting in a complexity of $O(N{\times}f)$. Then, in the main iteration of Table 2, the ranking process based on fitness values has a complexity of $O(N \log N)$, while the mutation operation applied to each individual has a complexity of $O(N)$. The fitness evaluation for all individuals is with a complexity of $O(N{\times}f)$, and identifying the optimal individual involves either sorting or traversing, adding a complexity of $O(N)$.

Consequently, the total complexity within the main iteration is

$$O(N \log N + N{\times}f) \tag{5}$$

Thus, the overall time complexity of the algorithm is expressed as

$$O(N) + O(N{\times}f) + O(MaxIter{\times}(N \log N + N{\times}f)) \tag{6}$$

After simplifying by neglecting lower-order terms, the time complexity is reduced to

$$O(MaxIter{\times}(N \log N + N{\times}f)) \tag{7}$$

Computational Experiments

Experimental Settings

To demonstrate the efficacy of our proposed hybrid method, some real polygon datasets were used via unmanned aerial vehicle oblique photogrammetry. First, 3D mesh tiles were generated through the

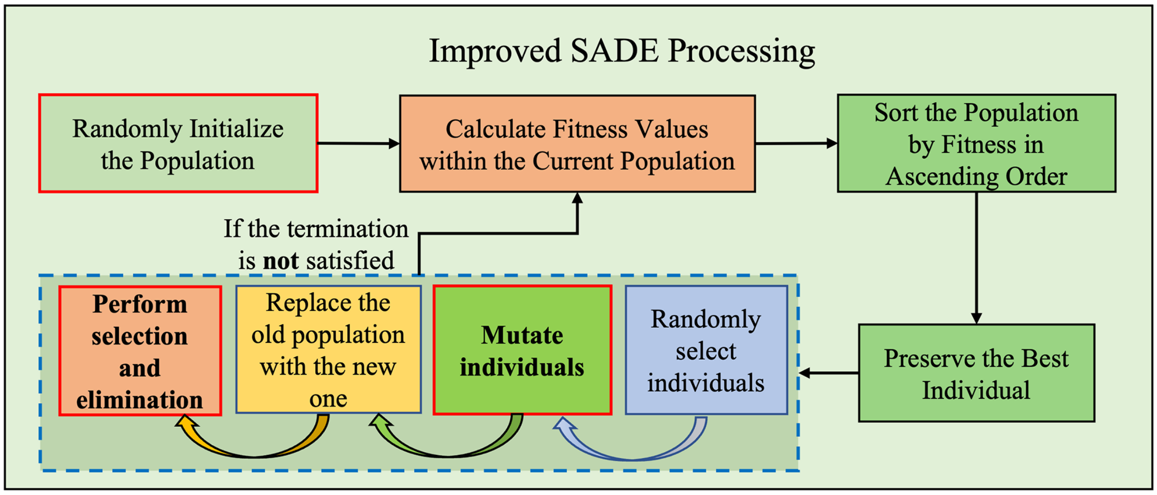

Improved SADE Processing
Randomly Initialize the Population
Calculate Fitness Values within the Current Population
Sort the Population by Fitness in Ascending Order
If the termination is **not** satisfied
Perform selection and elimination
Replace the old population with the new one
Mutate individuals
Randomly select individuals
Preserve the Best Individual

Figure 5. Improved simplified atavistic differential evolution (SADE) processing workflow.

commercial software ContextCapture, and the boundaries of individual texture maps were extracted from the relevant OBJ files and converted into polygonal coordinate data, forming the dataset for our testing.

For evaluation, three datasets of various sizes (D1, D2, and D3) were used to test our hybrid optimization method, each containing a different number of polygons. Based on these datasets, extensive ablation studies were reported; then, the traditional texture nesting methods based on GA and SA algorithm were compared. The details of the datasets are listed in Table 3 and sample images of experimental textures are shown in Figure 6. All the proposed methods were implemented in Python, and all tests were conducted on a MacBook equipped with an Apple M4 Pro chip and 24 GB of RAM. Code is available at https://github.com/louliyuan/NFP-SADE-With-Adaptive-Container-Resizing.

Table 3. Dataset characteristics.

Datasets	Total Number of Parts	Maximum Number of Vertices in a Polygon	Total Number of Vertices	Average Number of Vertices Per Polygon
D1	302	30	2276	7.54
D2	360	24	3544	9.84
D3	1123	30	8217	7.32

Ablation Studies

Performance of the Improved SADE Algorithm

In this section, the three datasets (D1, D2, and D3) are used to evaluate the proposed improved SADE algorithm. Table 4 reports the results of two metrics: execution time (the average of three independent trials for each dataset) and space use rates. The conducted experiments include a GA-based method (integrating GA with the traditional NFP approach), an SA-based method (integrating SA algorithm with the traditional NFP approach, referred to as *SA*), an original SADE-based method (integrating traditional SADE with the traditional NFP approach, referred to as *Original*), and an improved SADE-based method (integrating the improved SADE with the traditional NFP approach, referred to as *Improved*). It can be seen that the improved SADE algorithm consistently achieves the best execution times across all datasets. Notably, it achieves a 16.1% speed improvement on D1 and an 8.3% improvement on D3 compared with the original SADE method. Moreover, it demonstrates superior performance to both the GA and SA baselines, which are commonly used heuristic optimization techniques in polygon packing problems. In terms of space use, the improved SADE method maintains the high packing density of the original SADE algorithm (84.94%–85.46%), while outperforming the GA-based approach by a significant margin (up to 10% higher in D2). It is worth noting that while the SA-based method matches the packing rate of SADE, it falls short in computational efficiency.

Table 4. Performance of the improved simplified atavistic differential evolution. Time efficiency and space use rate are provided. Best result is highlighted in bold.

Method/Data	Execution Time (s)			Space Use (%)		
	D1	D2	D3	D1	D2	D3
	112.547	229.959	1418.115	65.43	75.73	83.61
GA	113.345	228.509	1373.179	65.43	85.46	82.56
	110.281	226.127	1439.077	65.43	85.46	82.73
average_1	112.057	228.198	1410.124	65.43	82.22	82.97
	111.565	219.710	1350.344	65.43	**85.46**	**84.94**
SA	112.896	215.375	1386.211	65.43	**85.46**	**84.94**
	109.694	220.550	1329.140	65.43	**85.46**	**84.94**
average_2	111.385	218.545	1355.232	65.43	**85.46**	**84.94**
	128.948	266.731	1446.152	65.43	**85.46**	**84.94**
Original	128.797	228.509	1463.868	65.43	**85.46**	**84.94**
	120.857	280.344	1516.427	65.43	**85.46**	**84.94**
average_3	126.201	275.967	1410.124	65.43	**85.46**	**84.94**
	109.518	**206.866**	**1305.766**	65.43	**85.46**	**84.94**
Improved	**108.754**	**207.889**	**1257.224**	65.43	**85.46**	**84.94**
	91.356	**204.347**	**1316.055**	65.43	**85.46**	**84.94**
average_4	**103.209**	**206.367**	**1293.015**	65.43	**85.46**	**84.94**

GA = genetic algorithm; SA = simulated annealing.

These results confirm that our improved SADE algorithm delivers a better balance between computational efficiency and packing quality, making it well suited for large-scale and time-sensitive applications

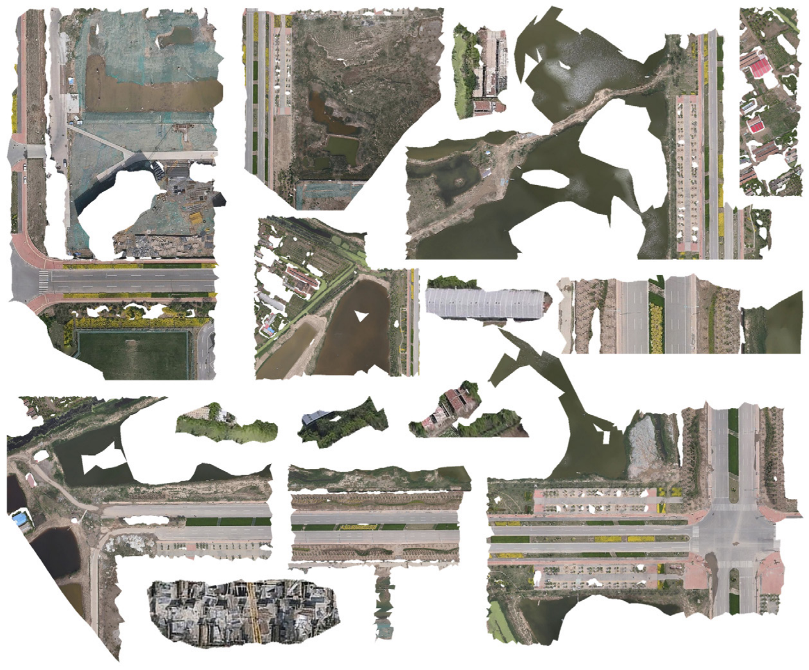


Figure 6. Sample textures of experimental data.

such as texture layout. Figure 7 further illustrates the qualitative differences among the GA-based, SA-based, and improved SADE-based placements, clearly highlighting the superior compactness and placement efficiency achieved by the proposed method.

Performance of NFP Parallelization

To explore the efficiency of the NFP parallelization strategy, it was incorporated into the *Improved* method, and the execution time was compared across three datasets. Each test was performed using a single iteration, and the final results were averaged over three independent trials, as summarized in Table 5. The results demonstrate that the NFP parallelization strategy improves computational efficiency by approximately 8% over the traditional NFP implementation.

Investigation of Iteration Number

Table 6 presents the results obtained using different numbers of iterations. The comparison is made between the Improved method executed with 2, 3, and 5 iterations. The results indicate that increasing the number of iterations typically has only a marginal effect on overall use of space, with no significant improvements observed. However, higher iteration counts cost substantially more execution time, reducing time efficiency. Note that this test assumes that the container size is sufficient to accommodate all polygons.

Table 5. Comparison of execution time (s) with and without no-fit polygon parallelization. Best data is highlighted in bold.

Method/Data	**D1**	**D2**	**D3**
	47.905	**93.793**	**649.874**
With	**47.477**	**99.568**	**647.443**
	50.087	**95.124**	**649.933**
average_time_1	**48.490**	**96.162**	**649.083**
	52.763	100.345	650.890
Without	52.436	99.773	656.647
	53.678	100.415	654.925
average_time_2	52.959	100.17748	654.154

Table 6. Comparison of space use and execution time across different iteration counts.

Counts/ Data	**Execution Time (s)**			**Space Use (%)**		
	D1	**D2**	**D3**	**D1**	**D2**	**D3**
2	103.209	206.367	1393.015	65.43	85.46	84.94
3	151.913	306.374	2015.819	65.43	85.46	84.94
5	254.660	508.987	3254.855	65.43	85.46	84.94

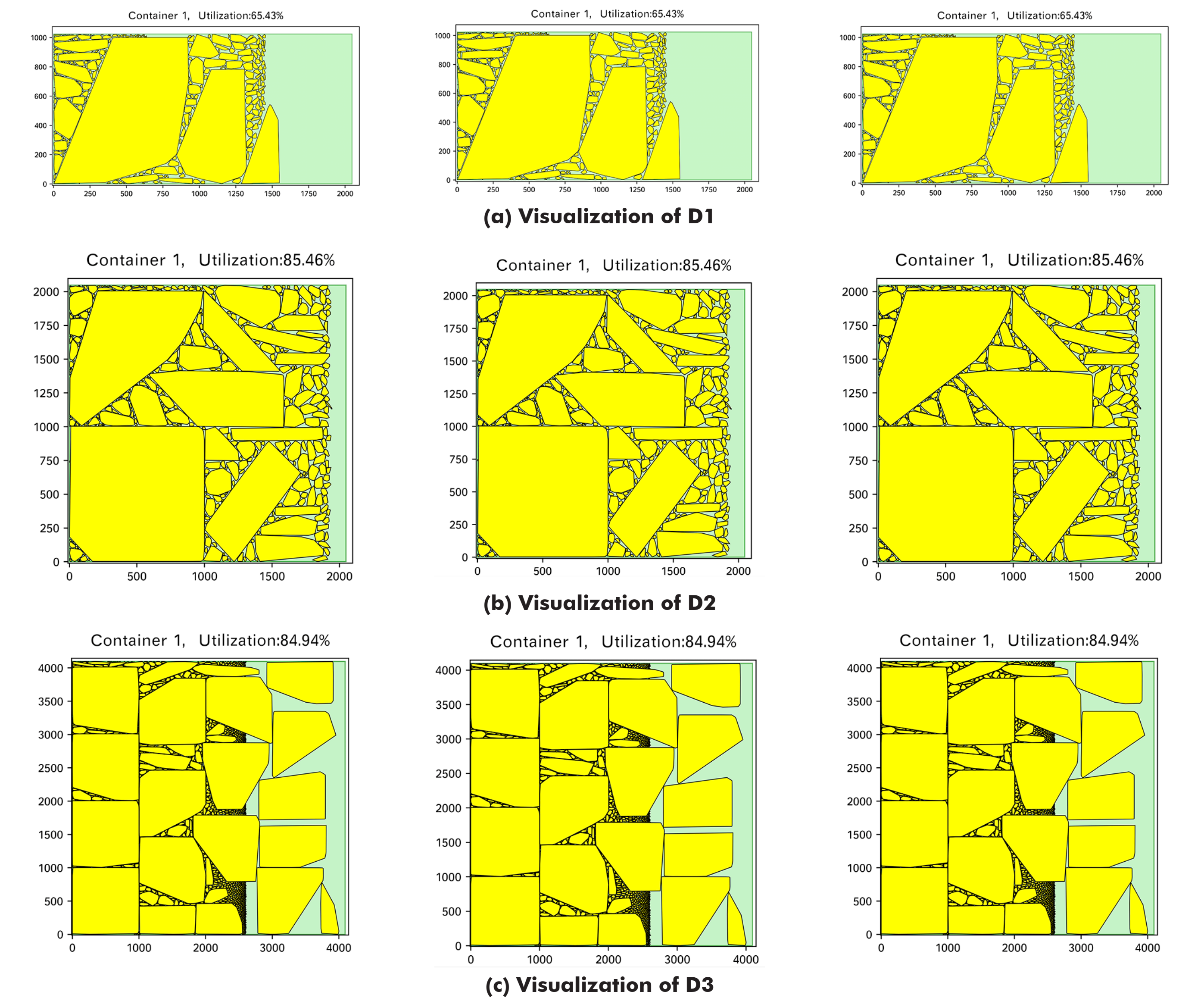


Figure 7. Comparison of visualization results among simplified atavistic differential evolution–based (left), genetic algorithm–based (middle), and simulated annealing–based (right) methods.

Performance of Container Group Strategy

To evaluate the effectiveness of the proposed container group strategy, Table 7 compares execution time and space use on different datasets, both with and without the container group strategy. The results show that integrating the container group strategy can significantly reduce execution time, achieving efficiency improvements of approximately 70% for D2 and D3. This demonstrates the superior performance of the optimized algorithm in handling large-scale datasets. Furthermore, use of space improved across all datasets, approaching *SUR* regardless of dataset size.

Table 7. Comparison of space use and execution time with and without container group (SUR = 95%). Best data is highlighted in bold.

Method/ Data	**Execution Time (s)**			**Space Use (%)**		
	D1	**D2**	**D3**	**D1**	**D2**	**D3**
	52.077	**41.551**	**427.984**	65.43	**92.02**	**87.45**
With	**45.086**	**41.615**	**428.817**	65.43	**92.02**	**87.45**
	47.169	**42.685**	**433.528**	65.43	**92.02**	**87.45**
Average	**48.110**	**41.950**	**430.110**	65.43	**92.02**	**87.45**
	109.518	206.866	1305.766	65.43	85.46	84.94
Without	108.754	207.889	1257.224	65.43	85.46	84.94
	91.356	204.347	1316.055	65.43	85.46	84.94
Average	103.209	206.367	1293.015	65.43	85.46	84.94

Tables 7 and 8 provide results using the threshold of *SUR* 95% and 90%, respectively. When space use requirement in NFP is lower, our optimized algorithm can run more quickly. Despite threshold (*SUR*) variations, the algorithm maintains a consistent performance trend across datasets, further confirming its reliability. *A* qualitative effect of the container group strategy on D2 and D3 is shown in Figure 8.

Table 8. Comparison of space use and execution time with and without container group (SUR = 90%). Best data is highlighted in bold.

Method/ Data	**Execution Time (s)**			**Space Use (%)**		
	D1	**D2**	**D3**	**D1**	**D2**	**D3**
	45.714	**37.404**	1256.504	65.43	**92.02**	84.94
With	**45.173**	**37.360**	1242.021	65.43	**92.02**	84.94
	44.987	**37.245**	1291.741	65.43	**92.02**	84.94
Average_1	**45.291**	**37.336**	1263.422	65.43	**92.02**	84.94
	109.518	206.866	1305.766	65.43	85.46	84.94
Without	108.754	207.889	1257.224	65.43	85.46	84.94
	91.356	204.347	1316.055	65.43	85.46	84.94
Average_2	103.209	206.367	1293.015	65.43	85.46	84.94

Overall, the optimized adaptive container resizing function demonstrates robust optimization performance under varying threshold (*SUR*) and dataset conditions. The strategy significantly enhances execution efficiency, reduces computation time, and improves use of space. These results confirm the robustness and stability of the proposed method. However, the observed performance differences across datasets highlight the critical influence of dataset characteristics (e.g., polygon count, size, and shape) on optimization effectiveness.

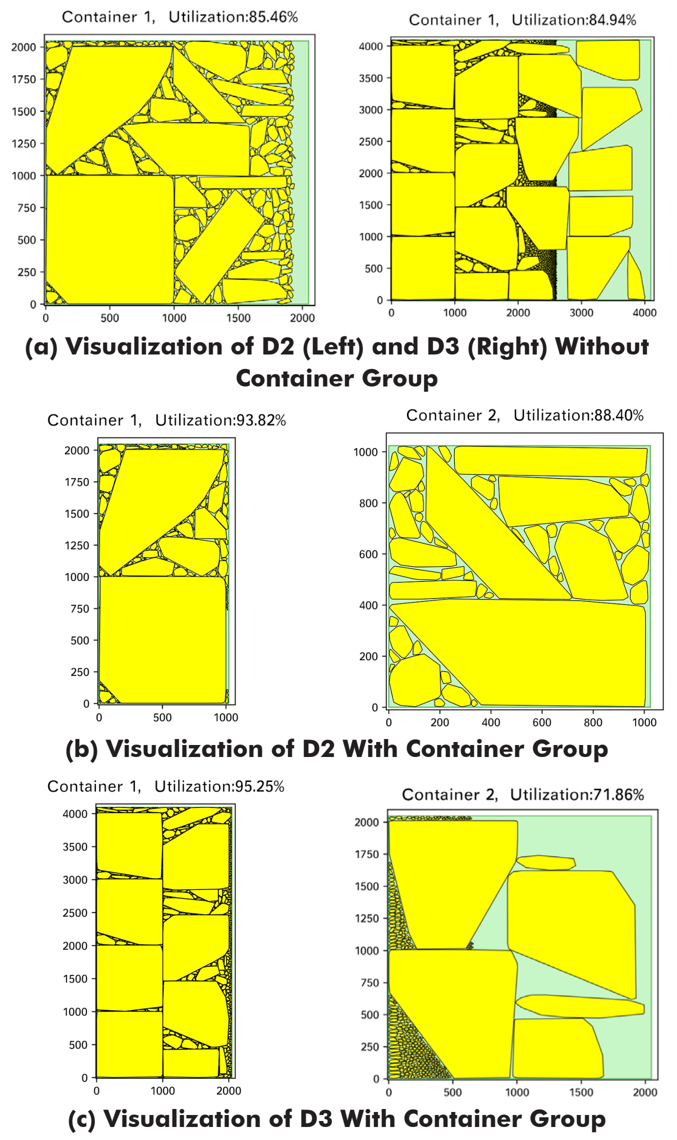
Comparison with Classical Heuristic Methods

In this section, the proposed method that integrates an enhanced NFP method with an improved SADE algorithm is compared with the classical GA-based method combining with the traditional NFP and the SA-based method that incorporates SA into the same layout framework. The results are presented in Table 9.

Table 9. Comparison of space use and execution time between the proposed and classical heuristic methods. Best data is highlighted in bold.

Method/ Data	**Execution Time (s)**			**Space Use (%)**		
	D1	**D2**	**D3**	**D1**	**D2**	**D3**
	52.077	**41.551**	**427.984**	65.43	**92.02**	**87.45**
Proposed	**45.086**	**41.615**	**428.817**	65.43	**92.02**	**87.45**
	47.169	**42.685**	**433.528**	65.43	**92.02**	**87.45**
Average	**48.110**	**41.950**	**430.110**	65.43	**92.02**	**87.45**
	112.547	229.959	1418.115	65.43	75.73	83.61
GA	113.345	228.509	1373.179	65.43	85.46	82.56
	110.281	226.127	1439.077	65.43	85.46	82.73
Average	112.057	228.198	1410.124	65.43	82.22	82.97
	111.565	219.710	1350.344	65.43	85.46	84.94
SA	112.896	215.375	1386.211	65.43	85.46	84.94
	109.694	220.550	1329.140	65.43	85.46	84.94
Average	111.385	218.545	1355.232	65.43	85.46	84.94

GA = genetic algorithm; SA = simulated annealing.



(a) Visualization of D2 (Left) and D3 (Right) Without Container Group

(b) Visualization of D2 With Container Group

(c) Visualization of D3 With Container Group

Figure 8. Visualization of different datasets before and after applying the container group strategy.

It can be found that the proposed method significantly outperforms both classical heuristic methods in terms of execution efficiency, achieving up to 5.44 times speed improvement over the GA-based method and 5.21 times speed improvement over the SA-based method, particularly for large-scale datasets such as D3. Additionally, the proposed method exhibits substantially higher use of space, indicating a more efficient algorithm design and a more effective resource allocation strategy. *A* sample irregular texture mapping result is shown in Figure 9, in which Figure 9a is the reorganized texture image and Figure 9b is the corresponding layout of irregular polygons.

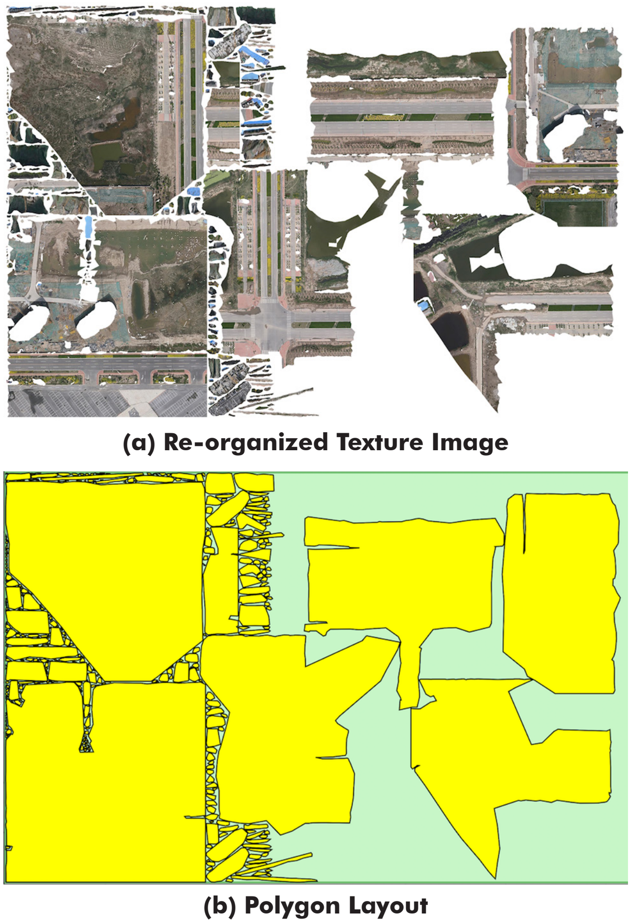


Figure 9. Visual results of reorganized texture image and polygon layout.

Conclusion

This paper presents a novel hybrid optimization approach for efficient irregular texture nesting, integrating an enhanced NFP method with an improved SADE algorithm. To address computational inefficiencies and suboptimal use of space in large-scale 3D texture nesting, we introduce a self-adaptive container resizing strategy, parallelized NFP processing, and refined mutation operations for polygon placement. These innovations collectively improve collision-free nesting exploration, maintain population diversity, and accelerate convergence.

Experimental results on real-world datasets demonstrate the superiority of the proposed method over two canonical heuristic methods, specifically the GA-based and SA-based methods. Without container grouping, our method outperforms both classical heuristic methods, achieving a 12% reduction in execution time and a 3%–10% improvement in use of space compared with the GA-based method. With adaptive container grouping, the execution time is reduced by up to 5.44 times over the GA-based method and 5.22 times over the SA-based method. Our method's practicability is validated by its consistent performance across datasets of varying sizes, exhibiting better space use rate and time efficiency. Additionally, extensive ablation studies prove the efficacy of parallelized NFP and adaptive container strategies in balancing time efficiency and space use rate.

Beyond the specific scope of texture layout optimization, the proposed approach demonstrates strong potential for broader applicability within urban informatics and smart city systems. In traffic management scenarios, heterogeneous vehicle types—represented as irregular polygons—can be dynamically configured in constrained urban spaces, such as intersections, parking zones, or emergency access routes. Optimizing these spatial configurations can enhance road usage efficiency, alleviate congestion, and improve safety and resilience. Furthermore, the algorithm's core mechanisms are transferable to domains such as logistics, intelligent warehouse systems, and virtual urban modeling, where scalable layout optimization and real-time spatial planning are essential.

In future work, we plan to extend the proposed framework to support more complex and dynamic polygonal structures, particularly those with internal holes. Although the current implementation focuses on external boundary–based NFP computation, interior voids are common in real-world texture data and can significantly affect nesting outcomes. Efficiently handling such features will require advancements in automatic hole detection and NFP generation tailored to hole-aware configurations. Additionally, we aim to incorporate support for nonconvex, dynamic datasets; explore the integration of deep learning techniques for predictive container initialization; and adopt GPU-accelerated parallelization to enhance scalability and computational performance. These directions represent key avenues for improving the flexibility, robustness, and real-time applicability of our hybrid optimization system.

Acknowledgement

This work was financially Supported by the National Nature Science Foundation of China (No. 42301507).

References

Achlioptas, P., Diamanti, O., Mitliagkas, I. and Guibas, L. 2018. Learning representations and generative models for 3D point clouds, *Proceedings of the 35th International Conference on Machine Learning*, 10–15 July 2018, Stockholm, Sweden (International Conference on Machine Learning: San Diego, California), pp. 40–49.

Aji, A., Hoang, V. and Wang, F. 2015. Effective spatial data partitioning for scalable query processing. arXiv:1509.00910 [cs.DB]. https://doi.org/10.48550/arXiv.1509.00910.

Bennell, J. A., Cabo, M. and Martinez-Sykora, A. 2018. A beam search approach to solve the convex irregular bin packing problem with guillotine cuts. *European Journal of Operational Research* 270(1), 89–102.

Bergmann, U., Jetchev, N. and Vollgraf, R. 2017. Learning texture manifolds with the periodic spatial GAN. arXiv:1705.06566 [cs.CV]. https://doi.org/10.48550/arXiv.1705.06566.

Charalambous, C. and Fleszar, K. 2011. A constructive bin-oriented heuristic for the two-dimensional bin packing problem with guillotine cuts. *Computers & Operations Research* 38(10):1443–1451.

Chen, Z. and Zhang, H. 2019. Learning implicit fields for generative shape modeling, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15–20 June 2019, Long Beach, California (IEEE: Piscataway, New Jersey), pp. 5939–5948.

Dai, X., Xiong, H. and Gong, J. 2015. A multrtexture automatic merging approach for the 3D city models. *Geomatics and Information Science of Wuhan University* 40(3):347–352+411.

Dean, H. T., Tu, Y. and Raffensperger, J. F. 2006. An improved method for calculating the no-fit polygon. *Computers & Operations Research* 33(6):1521–1539.

Dharma, K. C., Morrison, C. T. and Walls, B. 2022. Texture generation using a graph generative adversarial network and differentiable rendering, *International Conference on Image and Vision Computing New Zealand*, 24–25 November 2022, Auckland, New Zealand (Springer Nature: Cham, Switzerland), pp. 388–401.

Dumoulin, V., Shlens, J. and Kudlur, M. 2016. A learned representation for artistic style. arXiv:1610.07629 [cs.CV]. https://doi.org/10.48550/arXiv.1610.07629.

Efros, A. A. and Freeman, W. T. 2023. Image quilting for texture synthesis and transfer. In *Seminal Graphics Papers: Pushing the Boundaries*, vol. 2. Association for Computing Machinery: New York, New York, pp. 571–576.

Fan, W., Fang, J. and Huang, G. 2024. An improved image texture synthesis based on algorithm convolution neural network. *Physical Communication* 66:102395.

Gatys, L., Ecker, A. S. and Bethge, M. 2015. Texture synthesis using convolutional neural networks, *NIPS'15: Proceedings of the 29th International Conference on Neural Information Processing Systems*, July 10–15, 2018, Stockholm, Sweden, edited by J. Dy and A. Krause, vol. 1 (MIT Press: Cambridge, Massachusetts), pp. 262–270.

Ghosh, P. K. 1993. A unified computational framework for Minkowski operations. *Computers & Graphics* 17(4):357–378.

Gilmore, P. C. and Gomory, R. E. (1965). Multistage cutting stock problems of two and more dimensions. *Operations Research* 13(1), 94–120.

Hertz, A., Hanocka, R., Giryes, R. and Cohen-Or, D. 2020. Deep geometric texture synthesis. arXiv:2007.00074 [cs.GR]. https://doi.org/10.48550/arXiv.2007.00074.

Hrstka, O. and Kučerová, A. 2000. Search for optimization method on multidimensional real domains. *Contributions to Mechanics of Materials and Structures, CTU Reports* 4:87–104.

Hrstka, O., Kučerová, A., Lepš, M. and Zeman, J. 2003. A competitive comparison of different types of evolutionary algorithms[J]. *Computers & Structures*, 81(18-19): 1979-1990.

Hrstka, O. and Kučerová, A. 2004. Improvements of real coded genetic algorithms based on differential operators preventing premature convergence. *Advances in Engineering Software* 35(3–4):237–246.

Huang, P., Matzen, K., Kopf, J., Ahuja, N. and Huang, J. 2018. Deepmvs: Learning multi-view stereopsis, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 18–23 June 2018, Salt Lake City, Utah (IEEE: Piscataway, New Jersey), pp. 2821–2830.

Jiang, W., Guo, J. and Yang, J. 2016. A two-dimensional nesting algorithm by using no-fit polygon, *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 13-15 October 2016, Vancouver, Canada (IEEE: Piscataway, New Jersey), pp. 1–5.

Junior, B. A., Pinheiro, P. R. and Saraiva, R. D. 2013. Tackling the irregular strip packing problem by hybridizing genetic algorithm and bottom-left heuristic, *2013 IEEE Congress on Evolutionary Computation*, 20–23 June 2013, Cancun, Mexico (IEEE: Piscataway, New Jersey), pp. 3012–3018.

Kanazawa, A., Tulsiani, S., Efros, A. A. and Malik, J. 2018. Learning category-specific mesh reconstruction from image collections, *Proceedings of the European Conference on Computer Vision (ECCV)*, *Part XV*, 8–14 September 2018, Munich, Germany (SpringerNature), pp. 371–386.

Kang, K., Moon, I. and Wang, H. 2012. A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem. *Applied Mathematics and Computation* 219(3):1287–1299.

Kang, Z., Guan, Y., Wang, J. and Chen, P. 2024. Research on genetic algorithm optimization with fusion tabu search strategy and its application in solving three-dimensional packing problems. *Symmetry* 16(4):449.

Kipf, T. N. and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907 [cs.LG]. https://doi.org/10.48550/arXiv.1609.02907.

Lin, J., Xu, Z., Sharma, G. and Pappas, T. N. 2023. Texture representation via analysis and synthesis with generative adversarial networks. *e-Prime—Advances in Electrical Engineering, Electronics and Energy* 6:100286.

Ling, X. and Qin, R. 2023. Large-scale and efficient texture mapping algorithm via loopy belief propagation. *IEEE Transactions on Geoscience and Remote Sensing* 61:1–11.

Liu, C., Si, Z., Hua, J. and Jia, N. 2023. Optimizing two-dimensional irregular packing: a hybrid approach of genetic algorithm and linear programming. *Applied Sciences* 13(22):12474.

Liu, F., Shen, C., Lin, G. and Reid, I. 2015. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38(10):2024–2039.

Martinez-Sykora, A., Alvarez-Valdes, R., Bennell, J. and Oliveira, J. F. (2015). Constructive procedures to solve 2-dimensional bin packing problems with irregular pieces and guillotine cuts. *Omega* 52:15–32.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S. and Geiger, A. 2019. Occupancy networks: Learning 3d reconstruction in function space, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15–20 June 2019, Long Beach, California (IEEE: Piscataway, New Jersey), pp. 4460–4470.

Mumford-Valenzuela, C. L., Vick, J. and Wang, P. Y. 2004. Heuristics for large strip packing problems with guillotine patterns: An empirical study. In *Metaheuristics: Computer Decision-Making*, edited by M.G.C. Resende and J. P. de Sousa. Boston, Massachusetts: Kluwer Academic Publishers, pp. 501–522.

Niemeyer, M., Mescheder, L., Oechsle, M. and Geiger, A. 2020. Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14–19 June 2020, virtual (IEEE: Piscataway, New Jersey), pp. 3504–3515.

Park, T., Liu, M. Y., Wang, T. C. and Zhu, J. Y. 2019. Semantic image synthesis with spatially-adaptive normalization, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15–20 June 2019, Long Beach, California (IEEE: Piscataway, New Jersey), pp. 2337–2346.

Rouhani, M., Fradet, M., and Baillard, C. 2020. Efficient texture mapping via a non-iterative global texture alignment. arXiv:2011.00870 [cs.CV]. https://doi.org/10.48550/arXiv.2011.00870.

Sheng, X., Yuan, J., Tao, W., Tao, B. and Liu, L. 2021. Efficient convex optimization-based texture mapping for large-scale 3D scene reconstruction. *Information Sciences* 556: 143–159.

Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G. and Zollhofer, M. 2019. Deepvoxels: Learning persistent 3D feature embeddings, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15–20 June 2019, Long Beach, California (IEEE: Piscataway, New Jersey), pp. 2437–2446.

Stoyan, Y. G. and Ponomarenko, L. D. 1977. Minkowski sum and hodograph of the dense placement vector function. *Reports of the Ukrainian SSR Academy of Science,* ser. A, 10.

Sun, B., Li, G., Wang, S. and Xie, C. 2025. Two-dimensional bin-packing problem with conflicts and load balancing: A hybrid chaotic and evolutionary particle swarm optimization algorithm. *Computers & Industrial Engineering* 200:110851.

Teimury, F., Roy, B., Casallas, J. S., MacDonald, D. and Coates, M. 2020. Graphseam: Supervised graph learning framework for semantic UV mapping. arXiv:2011.13748 [cs.GR]. https://doi.org/10.48550/arXiv.2011.13748.

Toulatzis, V. and Fudos, I. 2021. Deep tiling: texture tile synthesis using a deep learning approach. arXiv:2103.07992 [cs.CV]. https://doi.org/10.48550/arXiv.2103.07992.

Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W. and Jiang, Y. G. 2018. Pixel2mesh: Generating 3D mesh models from single RGB images, *Proceedings of the European Conference on Computer Vision (ECCV)*, 8–14 September 2018, Munich, Germany (Springer, Cham, Switzerland) pp. 52–67.

Wang, X., Zhang, N., Wang, A. and Ge, Y. 2024. Research on 2D nesting optimization combining improved no-fit polygon method and genetic algorithm, *2024 IEEE International Conference on Mechatronics and Automation (ICMA)*, 4–7 August 2024, Tianjin, China (IEEE: Piscataway, New Jersey), pp. 62–67.

Wang, Y., Gu, Y., Miao, J., Zhang, Y., Jin, C. and Guan, G. 2024. Research on intelligent nesting algorithm for irregular ship parts based on no-fit-polygon. *Applied Ocean Research* 150:104108.

Wei, L. and Lim, A., et al. 2011. A skyline-based heuristic for the 2d rectangular strip packing problem, *Modern Approaches in Applied Intelligence: 24th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems* 28 June–1 July 2011, Syracuse, NY (Springer: Berlin, Germany), pp. 286–295.

Weinhaus, F. M. and Devarajan, V. 1997. Texture mapping 3D models of real-world scenes. *ACM Computing Surveys (CSUR)* 29(4):325–365.

Xian, W., Sangkloy, P., Agrawal, V., Raj, A., Lu, J., Fang, C., … and Hays, J. 2018. Texturegan: Controlling deep image synthesis with texture patches, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 18–23 June 2018, Salt Lake City, Utah (IEEE: Piscataway, New Jersey), pp. 8456–8465.

Xu, J., Wu, X., Liu, H. and Zhang, M. 2017. An optimization algorithm based on no-fit polygon method and hybrid heuristic strategy for irregular nesting problem, *2017 36th Chinese Control Conference (CCC)*, 26-28 July 2017, Dalian, China (IEEE: Piscataway, New Jersey), pp. 2858–2863.

Xu, R., Guo, M., Wang, J., Li, X., Zhou, B. and Loy, C. C. 2021. Texture memory-augmented deep patch-based image inpainting. *IEEE Transactions on Image Processing* 30: 9112–9124.

Xu, X., Zhou, L. 2023. Research on two-dimensional rectangular part layout problem based on improved genetic tabu search hybrid algorithm. *Operations Research and Fuzziology* 13:581.

Zhang, D., Shi, L. and Leung, S. C. H., et al. 2016. A priority heuristic for the guillotine rectangular packing problem. *Information Processing Letters* 116(1):15–21.

Zuo, Q., Liu, X. and Chan, W. K. V. 2022. A constructive heuristic algorithm for 3D bin packing of irregular shaped items, *INFORMS International Conference on Service Science*, 2–4 July 2022, Shenzhen, China (Cham, Switzerland: Springer International Publishing), pp. 393–406.