Graphcut-Based Animation Texture Synthesis With Temporal Consistency Constraints

Shuang Cui[1*] and Chunxin Zhang[2]

[1] School of Hunan Mass Media Vocational and Technical College; Changsha Hunan 410000 China
[2] College of Humanities, Xi'an Jiaotong University, Xi'an Jiaotong University City College, Xi'An 710018 Shaanxi, China
[*] Corresponding author. E-mail: peter15119576@163.com

Received: May 13, 2026; Accepted: July 28, 2026

Animated texture replacement is an important research problem in computer graphics and digital content generation. Its core challenge lies in simultaneously ensuring spatial continuity, temporal consistency, and lighting stability during texture stitching. Traditional texture synthesis methods are primarily designed for static images, and their direct application to animation sequences often leads to seam jitter, texture flickering, and lighting drift, thereby degrading visual quality. To address these limitations, this paper proposes a Graphcut-based animated texture replacement method. Within a block-level texture synthesis framework, the proposed approach constructs a pixel-level energy function and employs a Maximum Flow–Minimum Cut model to determine globally optimal seams in overlapping regions, thereby ensuring spatial consistency. Furthermore, temporal consistency constraints are introduced to suppress inter-frame texture fluctuations, while a customized lighting model maintains stable and controllable illumination throughout continuous animation sequences. To accelerate the computationally intensive block matching stage, CUDA-based parallel processing is incorporated, significantly improving execution efficiency. Experimental results demonstrate that the proposed method achieves an average Temporal Difference Error (TDE) of approximately 2.3, representing about a 61% reduction compared with the best-performing baseline method, while also achieving the lowest Temporal Illumination Variance (TIV), indicating superior temporal stability and illumination consistency. Overall, the proposed framework provides high-quality texture synthesis with improved computational performance, making it suitable for texture replacement tasks in complex animation scenes.

Keywords: Animated texture replacement; Graphcut algorithm; Temporal consistency; Lighting customization; Texture composition; CUDA parallel acceleration

http://dx.doi.org/10.6180/jase.202611_34.056

1. Introduction

With the rapid development of digital media technology and computer graphics, the demand for high-quality texture compositing and replacement technologies in fields such as animation production, film and television special effects, and virtual reality is increasing [1, 2]. Texture, as one of the most important visual elements in images and animation, directly affects the realism and artistic expression of a scene [3]. Therefore, how to achieve efficient, natural, and controllable texture replacement while ensuring visual quality has become an important research topic in the fields of computer graphics and digital content generation [4].

Early texture compositing research focused on static images, where sample-based methods generated arbitrary-sized textures by analyzing local texture features but often produced visible stitching artifacts [5, 6]. To address this limitation, image quilting introduced optimal seam searching in overlapping regions, while subsequent shortest-path

and dynamic programming-based seam optimization methods further improved texture stitching quality by reducing boundary discontinuities [7]. Building upon this foundation, the Graphcut algorithm, due to its globally optimal energy minimization characteristic, has been introduced into the fields of texture synthesis and image editing [8, 9]. By modeling overlapping regions as a graph and applying the maximum flow–minimum cut algorithm, the Graphcut method identifies the global minimum-energy seam, automatically avoiding high-discrepancy regions and significantly improving stitching naturalness [10]. As texture synthesis extends from static images to animation sequences, maintaining both spatial consistency and temporal stability becomes a major challenge. Independent frame-by-frame synthesis often causes temporal flickering, seam jumps, and texture wandering, degrading animation quality [11]. Existing video texture transfer methods, including PatchMatch and Poisson fusion, reduce stitching artifacts but still suffer from texture drift due to the lack of explicit temporal consistency constraints. In addition, most existing approaches assume constant lighting conditions and cannot effectively model dynamic illumination, resulting in brightness inconsistencies and lighting drift. Therefore, incorporating a controllable and stable illumination transfer mechanism is essential for improving the realism and quality of animation texture replacement [12]. Although Graphcut-based texture synthesis achieves high-quality results, its computational complexity limits efficiency in high-resolution animation. To address this, the proposed framework integrates Graphcut energy minimization, block-level texture synthesis, temporal consistency constraints, and a customized lighting model to suppress seam jitter and texture flickering, preserve target illumination and structural features, and improve visual realism and robustness under complex lighting conditions.

2. Material and methods

2.1. Texture Synthesis Method Based on Graph Cut Optimization

In texture synthesis tasks, to achieve natural and continuous stitching effects, the image reconstruction problem can be transformed into an energy minimization problem in graph theory. This method uses pixels as the basic unit, treating each pixel in the image as a node in a graph structure, and establishing edge connections between adjacent pixels based on spatial adjacency. Simultaneously, two virtual nodes—source node S and sink node T—are introduced, representing pixel affiliations from different texture regions, respectively [13, 14].

In the constructed graph model, each pixel node is not only connected to its neighboring pixels but also to S and T, forming a weighted directed graph. The overall optimization objective can be expressed as minimizing the following energy function:

$$E(L) = \sum_{p \in \Omega} D_p\left(L_p\right) + \sum_{(p,q) \in \mathcal{N}} V_{pq}\left(L_p, L_q\right) \tag{1}$$

Where: Ω represents the set of image pixels; $L_p \in \{S, T\}$ is the label of pixel p, indicating whether it belongs to the source texture or the target texture; $D_p\left(L_p\right)$ is a data term used to measure the matching error caused by pixel p selecting label L_p; N is the set of pixel adjacency pairs; $V_{pq}(\cdot)$ is a smoothing term used to penalize cases where adjacent pixels are assigned to different regions. In texture composition, data items are typically defined based on pixel color differences or local feature similarity, for example:

$$D_p(S) = \|I_1(p) - I(p)\|^2, D_p(T) = \|I_2(p) - I(p)\|^2 \tag{2}$$

Where I_1 and I_2 represent two textures to be stitched together, and $\mathrm{I(p)}$ is the current composite result. The smoothing constraint term can be written as:

$$V_{pq}\left(L_p, L_q\right) = \begin{cases} 0, & L_p = L_q, \\ \|I(p) - I(q)\|^2, & L_p \neq L_q. \end{cases} \tag{3}$$

By mapping the energy function to the weights of edges in the graph, the problem is ultimately transformed into finding the minimum cut from the source point S to the sink point T . The Maximum Flow-Minimum Cut algorithm models the overlapping region as a weighted graph to identify the globally optimal seam, avoiding high-discrepancy areas and ensuring smooth boundary transitions and structural continuity. During animation, temporal consistency aligns textures between consecutive frames using optical flow, while Graphcut computes the optimal seam and minimizes inter-frame appearance differences. This joint optimization suppresses seam jitter and texture flickering, preserves seamless texture transitions, and enables natural texture deformation with object motion while maintaining Graphcut-based spatial optimality.

2.2. Temporal Consistency Smoothing for Flicker Suppression

Let $T_t(x, y)$ denote the synthesized texture at frame t, and $T_{t-1}^{w}(x, y)$ represent the previous frame warped to the current frame using the estimated motion field. The temporally smoothed texture is computed as:

$$T_t^{\text{smooth}}(x, y) = \alpha T_t(x, y) + (1 - \alpha) T_{t-1}^{w}(x, y) \tag{4}$$

where $0 \leq \alpha \leq 1$ is the temporal smoothing coefficient controlling the balance between the current synthesized texture and the motion-compensated previous frame. A

larger α preserves current-frame details, whereas a smaller value increases temporal stability. The temporal consistency objective is formulated as:

$$E_{\text{temp}} = \sum_{(x,y)\in\Omega} \left\| T_t(x,y) - T_{t-1}^{w}(x,y) \right\|^2 \tag{5}$$

where Ω denotes the synthesized texture region. Minimizing E_{temp} reduces inter-frame intensity variations while preserving spatial texture structures. The final optimization combines spatial Graphcut energy with the temporal consistency constraint:

$$E_{\text{total}} = E_{\text{Graphcut}} + \lambda_t E_{\text{temp}} \tag{6}$$

where λ_t controls the contribution of temporal smoothing. This joint optimization suppresses texture flickering, seam jitter, and brightness fluctuations across consecutive animation frames while preserving the high-quality spatial stitching produced by the Graphcut algorithm. The Graphcut-based texture synthesis method can be broadly divided into two key stages: candidate block position search and optimal seam determination. In the candidate position matching stage, a scanline strategy is used to traverse the edges of the synthesized region, evaluating all possible overlapping placement positions. Based on the difference metric between the sample texture in the overlapping region and the current texture, the position with the smallest error is selected as the stitching candidate region [15]. Candidate overlap regions are evaluated using overlap similarity, and the region with the lowest matching error is selected, improving spatial alignment, block placement, and overall texture synthesis uniformity. The proposed framework integrates spatial continuity, temporal consistency, and lighting stability to minimize stitching artifacts, suppress seam jitter and texture flickering, maintain consistent illumination, and generate visually coherent, temporally stable, and realistic texture synthesis results.

2.3. Parallelization Optimization Strategy for Texture Synthesis Based on Runtime Bottleneck Analysis

Fig. 1 illustrates the elapsed time and runtime percentages for different processing stages across all procedures. Stage 1 dominates the execution time, contributing approximately 72–79 s of the total elapsed time, while Stage 2 and operations require only a small additional processing time, resulting in total execution times ranging from 76 to 83 s. The runtime percentages remain highly consistent, with Stage 1 varying between 1.51% and 1.75% and Stage 2 between 3.01% and 3.25%, indicating stable computational behaviour. Overall, the average execution time is approximately 79 s, demonstrating that the proposed approach maintains efficient and reliable performance with minimal variation across all procedures. Therefore, the proposed method selectively applies CUDA-based parallelization to the candidate matching stage, enabling concurrent evaluation of multiple candidate regions and significantly reducing overall processing time without compromising synthesis quality. Therefore, the overall performance bottleneck is concentrated in the search for the optimal placement of texture blocks. This stage requires repeated calculation of overlap errors over a large number of candidate regions, leading to a significant increase in computational load. Therefore, parallelization improvements for this module have the most significant acceleration potential. The overlap parameter directly influences seam quality, texture continuity, and computational cost in Graphcut-based texture synthesis. A larger overlap region provides more matching information, enabling lower-energy seam selection and improved boundary smoothness and structural continuity, but increases graph construction, optimization complexity, memory consumption, and runtime. Conversely, a smaller overlap reduces computational cost but may introduce visible stitching artifacts. GPU thread synchronization after each candidate distance computation ensures reliable result aggregation, balanced workload distribution, efficient resource utilization, and improved overall CUDA-based texture synthesis performance. After extensive evaluation on multiple texture samples, an overlap value of 36 pixels was selected as the optimal balance between visual quality and execution efficiency.

2.4. A Controllable Texture Synthesis Method Based on Illumination Constraints

The proposed method integrates an illumination constraint mechanism into the Graphcut framework by extracting local illumination from natural images and incorporating it into texture synthesis. This enables adaptive lighting modeling while preserving texture structure, resulting in improved brightness consistency, shadow preservation, lighting controllability, operational efficiency, and visual realism under specified illumination conditions.

Fig. 2 illustrates the controlled-light texture synthesis process. The algorithm first identifies the target region for texture replacement and estimates the local illumination distribution from a reference image. The extracted lighting information is then incorporated into the texture synthesis process, jointly guiding texture block selection and stitching to preserve structural continuity while matching the specified lighting pattern. This approach enables realistic, controllable texture replacement with consistent illumination throughout the synthesized region.

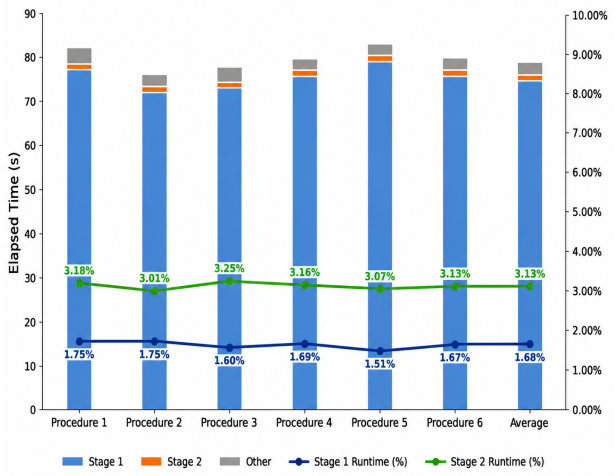

Fig. 1. Statistical chart of running time for each stage.

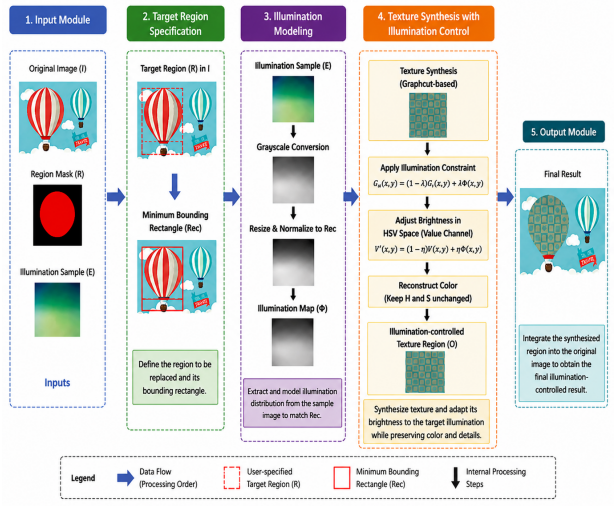

Fig. 2. Flowchart of the lighting customization algorithm.

1. Target Region Determination

   The first step in controlled illumination texture synthesis is to determine the target region in the original image I that needs to be replaced. This region is usually marked by the user using a binary mask to indicate the location of the texture to be synthesized. As shown in Fig. 2, the mask region corresponds to the part of the image where the lighting effect is to be regenerated. Using this mask, a set of target regions can be obtained.

2. Illumination Distribution Modeling

   To transfer the illumination features from the reference image to the target region, it is necessary to first estimate the illumination distribution from the illumination sample image E.

3. Texture Composition under Illumination Constraints

   After obtaining the reference illumination map Φ, it needs to be fused into the texture composition process of the target region. Let the final output region be O, where the region to be replaced satisfies

$$L_O(x,y) = f\left(L_I(x,y), \Phi(x,y)\right) \tag{7}$$

Where $L_I(x,y)$ is the grayscale value of the original image at position (x, y), and the function $\underline{f}(\cdot)$ is used to adjust the weight relationship between texture details and target lighting. A common form can be written as

$$L_O(x,y) = \lambda\Phi(x,y) + (1-\lambda)L_I(x,y), 0 \leq \lambda \leq 1 \tag{8}$$

The parameter λ controls the intensity of the light migration. When λ is large, the output region tends to be closer to the reference lighting distribution; when it is small, more original texture features are preserved. The illumination mapping function adaptively transfers the reference lighting characteristics to the synthesized texture while preserving its underlying structural and texture features. By integrating adaptive brightness compensation with the lighting model, the proposed method maintains illumination consistency across consecutive frames, facilitates natural visual transitions, and improves long-term rendering stability throughout the animation sequence. After estimating the illumination features of the target region, the next step is to perform texture synthesis within that region and further apply illumination transformations to generate the final output. First, a structurally continuous synthetic texture region is obtained based on the Graphcut texture stitching strategy. Then, its brightness distribution is adjusted using a lighting mapping model to conform to the pre-defined target illumination conditions. The output region of this process is denoted as O. The proposed illumination mapping framework estimates the local illumination distribution and applies illumination transfer only to the segmented target region. For irregular regions, illumination is generated within the minimum bounding region and masked according to the object boundary, preserving the original lighting pattern without affecting surrounding areas. Combined with Graphcut seam optimization, it maintains structural continuity and smooth illumination transitions across region boundaries. Brightness adaptation is achieved by modifying only the Value (V) component in the HSV color space while preserving the Hue (H) and Saturation (S) components, ensuring color consistency and visual fidelity. Unlike conventional methods that rely on fixed brightness adjustment or frame-independent illumination mapping, the proposed approach adaptively transfers local lighting characteristics while preserving texture structure.

Together with temporal consistency constraints, it maintains stable brightness across consecutive frames, reduces lighting flicker and drift, and produces more realistic texture appearance under dynamically changing illumination conditions. In addition to visual quality assessment, this paper also conducted a statistical analysis of the algorithm's runtime, the results of which are summarized in Table 1. Experiments show that the overall computational cost did not increase significantly after introducing the lighting customization module. Even under high-resolution image conditions, the time consumed by the illumination transfer part is only about 12% of that of the original Graphcut texture synthesis algorithm, indicating that the method can maintain high-quality synthesis results while incurring only a small additional computational cost. Each experiment was repeated 10 independent times under identical hardware and software configurations.

2.5. CUDA-Based Acceleration Strategy for Key Stages of Texture Synthesis

The proposed CUDA acceleration focuses on the texture block position evaluation module by parallelizing the distance computation of overlapping texture candidates. A hybrid CPU–GPU architecture is adopted, where the CPU manages synthesis iterations, memory allocation, and kernel scheduling, while the GPU performs large-scale parallel evaluation of candidate texture blocks. By distributing computations across thousands of GPU threads, the method significantly reduces the Graphcut matching bottleneck, improves throughput, efficiently processes high-resolution textures, and substantially decreases overall rendering time while maintaining synthesis quality. Therefore, the proposed framework adopts a CPU–GPU collaborative architecture, where the CPU manages scheduling, sequentially launches GPU parallel kernels, and updates synthesis results after each iteration to maintain structural and visual consistency throughout the texture synthesis process. The proposed CUDA-based optimization reconstructs the overlap distance computation of the original texture synthesis algorithm through parallel processing. Two CUDA parallelization strategies are proposed. The first accelerates single candidate evaluation by parallelizing pixel-level overlap distance calculations on the GPU, while the second performs simultaneous evaluation of multiple candidate positions, with each thread block processing one or more candidate regions to improve overall throughput. These strategies represent pixel-level and candidate-level parallelism, respectively, and Fig. 3 illustrates the CPU–GPU scheduling, GPU kernel execution, and integration with subsequent texture stitching.

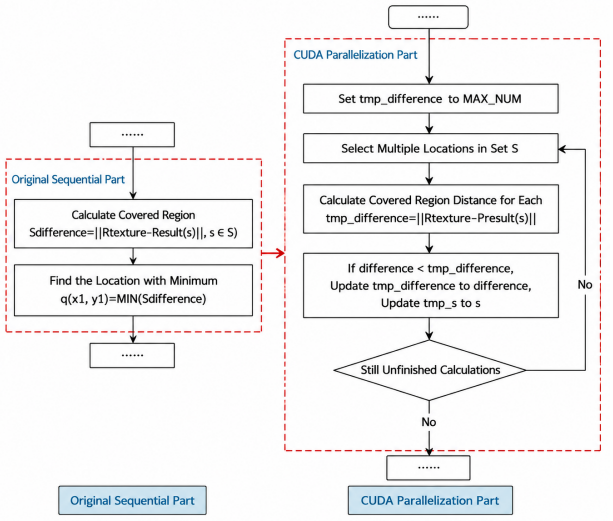

Fig. 3. The original CPU step flow area (left) and the enhanced parallelized step flow region (right) are depicted.

3. Results and discussion

The proposed interactive animation texture replacement system integrates image segmentation, automatic texture recommendation, and texture synthesis into a unified workflow. A large-scale texture database containing approximately 15,000 high-quality 96 × 96 texture samples, collected from multiple image sources and categorized by material type, supports diverse creative applications and accurate texture replacement. Figure 4 shows single-region texture replacement results, where semantic texture recommendation provides multiple candidate schemes, enabling users to generate visually appealing textures with artistic quality comparable to the original images. The proposed framework enables efficient multi-region texture replacement through semantic texture recommendation and a material-based texture database. By integrating seam optimization, illumination mapping, spatial continuity, and temporal coherence, it preserves structural continuity and lighting consistency, suppresses stitching artifacts, texture drift, and flickering, and improves the visual quality and perceptual realism of animation texture synthesis.

Fig. 4 presents keyframes (Frames 1, 10, 20, and 40) from a continuous animation sequence, demonstrating that the proposed method maintains stable texture structure during character motion. The Graphcut optimal seam search preserves spatial continuity by avoiding high-discrepancy regions, while the customized lighting model and temporal consistency constraints maintain stable brightness distribution across consecutive frames, effectively suppressing seam jumps, texture drift, flickering, and light-

Table 1. Runtime comparison under different output resolutions (unit: seconds).

Output Size	Original Graphcut Time (s)	Lighting Customization Time (s)	Total Time (Proposed) (s)	Lighting Ratio (%)
512 × 512	8.42 ± 0.11	0.91 ± 0.03	9.33 ± 0.12	10.8
768 × 768	15.76 ± 0.19	1.88 ± 0.05	17.64 ± 0.20	11.9
1024 × 1024	28.35 ± 0.28	3.47 ± 0.07	31.82 ± 0.30	12.2
1280 × 1280	46.10 ± 0.41	5.92 ± 0.09	52.02 ± 0.43	12.8

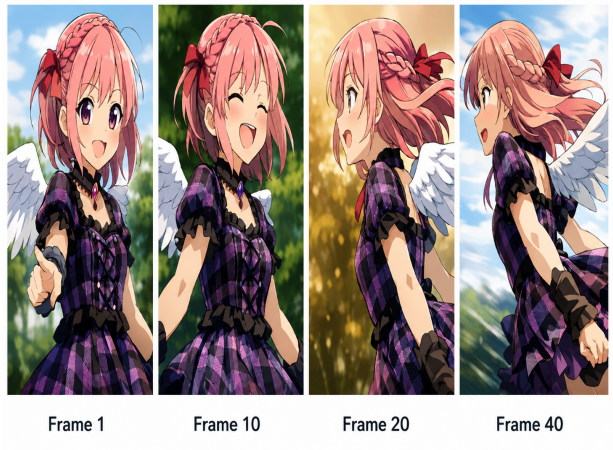

Fig. 4. Experimental results of texture replacement in animation sequences.

ing drift. Fig. 5 compares the Temporal Difference Error

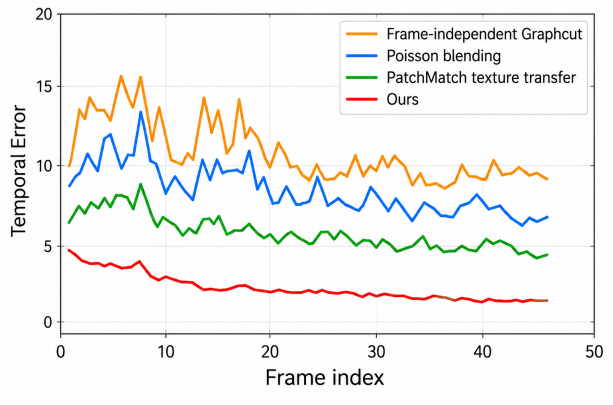

Fig. 5. Temporal Differential Error (TDE) of different methods in animation sequences.

(TDE) of the proposed method with Graphcut, Poisson fusion, and PatchMatch. TDE, computed using optical flow-based alignment and pixel-wise brightness differences between consecutive frames, measures inter-frame consistency, where lower values indicate better temporal stability. The Graphcut, Poisson fusion, and PatchMatch methods exhibit higher TDE values due to texture jumps, brightness drift, and texture wandering, respectively. In contrast, the proposed method maintains the lowest and smoothest TDE curve throughout the animation sequence, demonstrating superior temporal coherence, reduced texture flickering, seam instability, and brightness fluctuations during continuous animation generation.

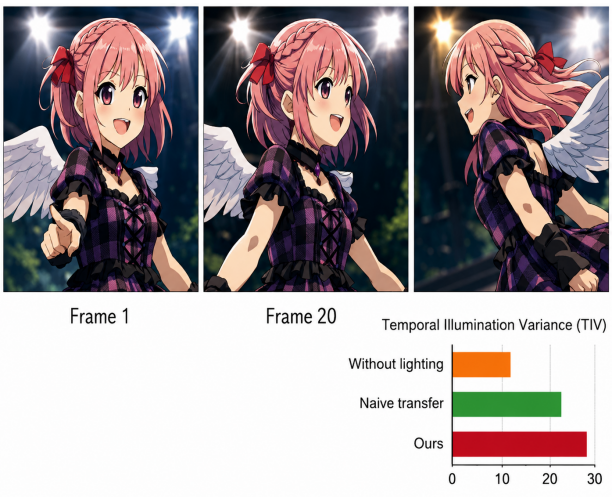

Fig. 6. Dynamic lighting consistency in an animation sequence.

Fig. 6 demonstrates the dynamic lighting preservation capability of the proposed method using Temporal Illumination Variance (TIV), where lower TIV values indicate better lighting consistency. Methods without lighting constraints exhibit significant brightness drift, while traditional lighting transfer methods still produce temporal fluctuations during animation. In contrast, the proposed method maintains stable brightness distribution across consecutive frames, ensuring consistent illumination without flickering or sudden brightness changes, and achieves the lowest TIV value compared with the no-light control and traditional light migration methods. Temporal Illumination Variance (TIV) is quantified as the variance of the HSV Value (V) channel across consecutive animation frames to measure frame-to-frame illumination fluctuations. Lower TIV values indicate reduced brightness drift and more stable lighting transitions, demonstrating the effectiveness of the proposed illumination adaptation strategy in maintaining consistent visual appearance throughout the animation

sequence.

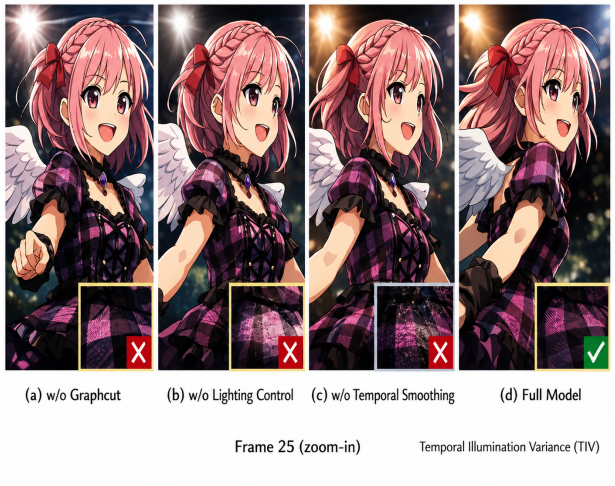

Fig. 7. Ablation experiment of animation texture replacement method.

Fig. 7 presents the ablation study of the proposed animation texture replacement method by disabling the Graphcut seam optimization, lighting control, and temporal consistency modules individually. Removing Graphcut causes seam discontinuities and structural damage, removing lighting control leads to brightness inconsistency and lighting drift, and removing temporal consistency results in inter-frame flickering and unstable texture details. In contrast, the complete model integrates Graphcut optimal seams, customized lighting mapping, and temporal consistency constraints to maintain stable texture structure, consistent illumination, and effectively suppress seam jumps, brightness drift, and temporal flicker.

Analysis of Individual Module Contributions: The ablation study demonstrates the complementary roles of the three proposed modules. The Graphcut optimization module primarily improves seam smoothness and structural continuity by identifying globally optimal stitching paths. The lighting customization module preserves consistent brightness distribution across synthesized textures under varying illumination conditions, while the temporal consistency module minimizes inter-frame texture variations to suppress flickering and texture drift. The complete framework, integrating all three modules, achieves the best overall performance by simultaneously maintaining seamless texture boundaries, stable illumination, and high temporal coherence throughout the animation sequence.

Fig. 8 compares seam locations produced by the direct rectangular stitching, DP-based seam search, and the proposed Graphcut seam optimization methods. The direct stitching method generates straight seams that often

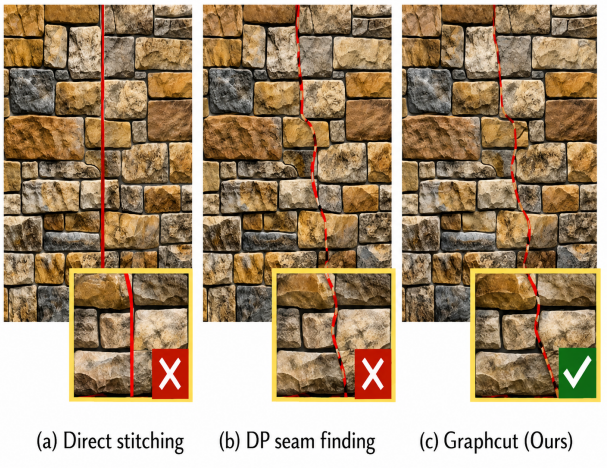

Fig. 8. Visualization of seams using different texture splicing methods.

cross significant texture changes, resulting in visible stitching boundaries. The DP-based method reduces this issue but may still create local discontinuities due to its limited search direction. In contrast, the proposed Graphcut method uses a pixel-level energy function and a maximum flow–minimum cut model to globally identify the minimum-energy seam, enabling seams to avoid major texture variations and achieve smoother, more natural texture transitions.

Memory Consumption Analysis: The proposed Graphcut seam optimization requires slightly higher memory than the DP-based method due to its pixel-level graph construction. However, the additional memory is confined to the local overlapping region, making it practical for high-resolution texture synthesis while providing superior seam quality, spatial continuity, and structural preservation.

Practical Deployment Limitations: The CUDA-based acceleration framework significantly reduces computational cost but relies on CUDA-compatible NVIDIA GPUs with sufficient memory and parallel resources. Performance may be affected by CPU–GPU data transfer, GPU memory limitations, and synchronization overhead. Nevertheless, it provides an effective balance between computational efficiency and synthesis quality, making it well suited for offline animation rendering and high-performance production pipelines.

4. Conclusion

This paper addresses common problems in animation texture replacement, such as discontinuous seams, temporal flicker, and illumination drift, and proposes a graphcut-based animation texture replacement method. By con-

structing a pixel-level energy function and introducing a maximum flow-minimum cut model, a globally optimal seam search in overlapping regions is achieved, providing a stable spatial structure foundation for texture synthesis. Furthermore, this paper incorporates temporal consistency constraints to effectively suppress texture jitter and flicker in consecutive animation frames; simultaneously, a customized illumination model is introduced to ensure that the synthesized texture maintains a stable and controllable brightness distribution throughout the animation sequence. To address the high computational complexity of the algorithm, this paper designs and implements a CUDA-based parallel acceleration scheme, significantly improving operational efficiency while maintaining synthesis quality. Experimental results show that the proposed method outperforms several comparative methods in terms of temporal difference error and illumination stability, achieving high-quality and stable texture replacement effects in complex animation scenes. Future work will further explore the integration of more complex illumination models and learning-based methods to enhance the algorithm's adaptability in real-world dynamic scenes.

Conflict of interest

The authors declare that they have no conflicts of interest regarding this work.

Data availability

The data that support the findings of this study are not publicly available due to confidentiality agreements but are available from the corresponding author upon reasonable request.

Author contributions

Shuang Cui and Chunxin Zhang, contributed to the design and methodology of this study, the assessment of the outcomes, and the writing of the manuscript.

References

[1] K. Zhang, D. Zhu, X. Min, and G. Zhai. "Textured Mesh Saliency: Bridging Geometry and Texture for Human Perception in 3D Graphics". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. ***39***. 9. 2025, 9977–9984. DOI: 10.1609/aaai.v39i9.33082.

[2] P. Tu, L.-Y. Wei, and M. Zwicker. "Compositional Neural Textures". In: *SIGGRAPH Asia 2024 Conference Papers*. New York: ACM, 2024, 1–11. DOI: 10.1145/3680528.3687561.

[3] H. Wu, Z. Wang, Y. Li, X. Liu, and T.-Y. Lee, (2024) *"Suitable and Style-Consistent Multi-Texture Recommendation for Cartoon Illustrations"* **ACM Transactions on Multimedia Computing, Communications and Applications *20***(7): 1–26. DOI: 10.1145/3652518.

[4] Y. Zeng, L. Ma, and P. V. Sander. "GSWT: Gaussian Splatting Wang Tiles". In: *SIGGRAPH Asia 2025 Conference Papers*. New York: ACM, 2025, 1–11. DOI: 10.1145/3757377.3763967.

[5] X. Liang, W. Yao, X. Fang, and C. Zhang, (2025) *"FMIF: Facial Multi-Feature Information Fusion for Driver Fatigue Detection"* **Signal, Image and Video Processing *19***(2): Article 121. DOI: 10.1007/s11760-024-03573-8.

[6] Z. Ou, X. Liu, C. Li, Z. Wen, P. Li, Z. Gao, and H. Wu, (2024) *"Body Part Segmentation of Anime Characters"* **Computer Animation and Virtual Worlds *35***(6): e2295. DOI: 10.1002/cav.2295.

[7] T. Elsner, J. Berger, T. Wu, V. Czech, L. Gao, and L. Kobbelt. "Retargeting Visual Data with Deformation Fields". In: *European Conference on Computer Vision*. Cham: Springer, 2024, 271–288. DOI: 10.1007/978-3-031-72949-2_16.

[8] W. Gong and Q. Hou, (2026) *"Advanced Techniques in Digital Media Processing for Special Effects Enhancement in Film and Television Post-Production"* **Multimedia Systems *32***(1): 17. DOI: 10.1007/s00530-025-02077-w.

[9] I. Oliveira, T. Pinto, S. Afonso, M. Karaś, U. Szymanowska, B. Gonçalves, and A. Vilela, (2025) *"Sustainability in Bio-Based Edible Films, Coatings, and Packaging for Small Fruits"* **Applied Sciences *15***(3): 1462. DOI: 10.3390/app15031462.

[10] M. Vijendran, J. Deng, S. Chen, E. S. L. Ho, and H. P. H. Shum, (2024) *"Artificial Intelligence for Geometry-Based Feature Extraction, Analysis and Synthesis in Artistic Images: A Survey"* **Artificial Intelligence Review *58***(2): Article 64. DOI: 10.48550/arXiv.2412.01450.

[11] J. Ma, E. Lu, R. Paiss, S. Zada, A. Holynski, T. Dekel, and F. Cole. "VidPanos: Generative Panoramic Videos from Casual Panning Videos". In: *SIGGRAPH Asia 2024 Conference Papers*. New York: ACM, 2024, 1–11. DOI: 10.1145/3680528.3687664.

[12] M. Xu, X. Tao, J. Liu, L. Pan, and Z. Huang, (2025) *"Research on the Innovation of Rattan Weaving Art Design Form and Inheritance of Cultural Symbols of Maonan Flower Bamboo Hat Based on Complex Geometry and AI Algorithm"* **Journal of Combinatorial Mathematics**

and Combinatorial Computing *127*: 9365–9383. DOI: 10.61091/jcmcc127a-522.

[13] G. Zou, J. Jiang, and Q. Chen, (2024) *"Accelerated Reconstruction of Scenes Using CUDA-Based Parallel Computing"* **IEEE Access *13***: 10489–10498. DOI: 10.1109/ACCESS.2024.3523099.

[14] M. Li, Z. Bi, T. Wang, Y. Wen, Q. Niu, X. Song, and M. Liu, (2024) *"Deep Learning and Machine Learning with GPGPU and CUDA: Unlocking the Power of Parallel Computing"* **arXiv**: DOI: 10.48550/arXiv.2410.05686.

[15] G. Mellone, C. G. De Vita, E. Di Nardo, G. Coviello, D. Di Luccio, P. P. C. Aucelli, and R. Montella. "G-Litter Marine Litter Dataset Augmentation with Diffusion Models and Large Language Models on GPU Acceleration". In: *2025 33rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*. 2025, 526–535. DOI: 10.1109/PDP66500.2025.00081.