Postprint of Texture-Adaptive Interpolation Method Based on Poisson Filling

Authors: Zhang Jun, Chen Kaiwen

Date: 2019-01-28T00:00:00+00:00

Abstract

To address texture synthesis problems under new graphics technology conditions, we propose a texture interpolation method that can adaptively stretch source texture images to different sizes while preserving their sharpness. First, a high-dimensional image interpolation algorithm is employed to scale the source texture to the target resolution, serving as an intermediate transitional texture. Second, by exploiting the self-similarity property of natural images, pixel patches are randomly selected from the source texture based on the features of intermediate texture pixels. Finally, the Poisson image editing algorithm is utilized to smoothly embed the source texture patches into the gap regions of the intermediate texture, producing the final synthesized texture. Extensive comparative experiments with existing algorithms demonstrate that the proposed method is applicable to both static and non-static texture synthesis problems, and the synthesis results achieve high visual consistency with the source texture. Additionally, the algorithm features simple logic and computational efficiency, without requiring complex optimization calculations or learning/training steps, making it suitable for deployment on mobile platforms with low hardware configurations.

Full Text

Preamble

Vol. 37 No. 4
Application Research of Computers
ChinaXiv Partner Journal

Adaptive Texture Interpolation by Poisson Filling

Zhang Jun a,b, Chen Kaiwen a
(a. School of Digital Media; b. Jiangsu Key Laboratory of Media Design &

Software Technology, Jiangnan University, Wuxi, Jiangsu 214122, China)

Abstract: Addressing texture synthesis challenges under new graphics technology conditions, this paper proposes a texture interpolation method that adaptively stretches source texture images to different sizes while preserving their clarity. First, a high-dimensional image interpolation algorithm is employed to split the source texture into the target resolution, generating an intermediate guidance texture. Second, leveraging the self-similarity of natural images, pixel patches are randomly selected from the source texture based on the characteristics of intermediate texture pixels. Finally, the Poisson image editing algorithm seamlessly embeds these source texture patches into the gap regions of the intermediate texture to produce the final synthesized texture. Extensive comparative experiments with existing algorithms demonstrate that the proposed method applies to both stationary and non-stationary texture synthesis problems, yielding results with high visual consistency relative to the source texture. Additionally, the algorithm features simple logic and fast computation without requiring complex optimization or learning-based training steps, making it suitable for mobile platforms with low hardware configurations.

Keywords: texture synthesis; high-dimensional interpolation; Poisson filling; non-stationary texture

0 Introduction

In recent years, the film and gaming industries have demanded increasingly realistic computer graphics, imposing ever-higher standards for visual authenticity in 3D virtual scenes. However, graphics hardware limitations prevent the industry from further enhancing scene realism by increasing geometric mesh complexity or lighting calculation sophistication. Consequently, the only viable approach is to extensively employ high-quality texture maps to indirectly augment scene realism.

A prominent issue with texture mapping is the frequent need for different image sizes when facing varying display areas. To avoid storing numerous high-resolution images, texture synthesis technology has gained widespread industry acceptance by enabling dynamic generation of required textures from a small number of examples or parameters. As a result, substantial research on texture synthesis techniques exists in current literature, achieving remarkable success in computer graphics applications [1, 2]. Mainstream texture synthesis algorithms primarily adopt a patch-based matching approach, where sub-regions are selected from example images according to various criteria and copied to specific positions in the target texture [3–10]. To ensure synthesized textures appear free of repetition and seams, numerous algorithms have been proposed to optimize patch selection or arrangement [11–15]. These methods treat the texture patch library as a variable domain and define the splicing effect as an error energy function, thereby reducing texture synthesis to a traditional optimization

problem of minimizing energy. This approach has achieved favorable results for small-sized examples and highly repetitive stationary textures, generating large-scale textures with minimal artificial artifacts.

For high-resolution non-stationary texture samples, whose geometric features exhibit spatially heterogeneous distributions across multiple scales, existing mainstream texture synthesis techniques cannot seamlessly stitch all features into the target texture. To address this challenge, researchers have recently proposed solutions involving guidance maps [16] and deep learning frameworks [17-19]. However, current guidance map methods [16] involve complex nonlinear dimensionality reduction calculations, resulting in unacceptably slow computation speeds when handling high-resolution example textures. While adversarial deep learning methods [18] can rapidly generate high-resolution textures, their training processes are time-consuming, and the trained networks are massive, requiring substantial memory to store training datasets and results—making them impractical for mobile platforms. Moreover, once deep learning algorithms complete training, they lack adjustable parameters to meet diverse texture synthesis needs, requiring retraining with updated example libraries and thus lacking flexibility.

In practice, with advances in computer hardware and image acquisition technology, the demand for synthesizing infinitely large artificial textures has gradually decreased. Instead, many application scenarios involve scaling or stretching existing high-resolution textures to adapt to different screen sizes (mobile phones, tablets, VR/AR devices, wearable devices, PCs, etc.). To ensure visual consistency of virtual scenes in computer games and animated films across different devices, synthesized textures must maintain consistent visual features and clarity (dots per inch, DPI) with the source texture.

This paper frames this new texture synthesis requirement as a texture interpolation problem and proposes a high-dimensional patch interpolation algorithm to generate intermediate textures, followed by Poisson image editing to progressively fill gaps between pixel blocks. The method applies to adaptive scaling of non-stationary textures, synthesizing textures at specified resolutions while preserving source texture feature distribution and DPI (as shown in Figure 1 [Figure 1: see original paper]). By avoiding complex optimization or learning-based approaches, the algorithm achieves linear time complexity O(N) and offers advantages of simplicity, ease of programming, and low hardware requirements.

It should be particularly noted that traditional image retargeting [20-22] or image super-resolution [23, 24] methods are unsuitable for texture interpolation problems. Existing image retargeting methods are essentially adaptive cropping algorithms with limited upscaling capability, while image super-resolution methods severely affect the DPI of source textures, causing visual quality degradation.

1 Algorithm Framework

This section introduces the overall workflow of the Poisson filling-based texture interpolation method, with theoretical details elaborated in Section 3. Example-based texture synthesis techniques can be broadly categorized into patch-based and pixel-based methods. Since patch-based synthesis maintains DPI consistency between results and example images, it is better suited for large-scale texture synthesis requirements.

The main operation of example-based patch synthesis involves selecting appropriate patches (e.g., 8×8 regions) from the example image and writing them to specific positions in the synthesized texture (as shown in Figure 2 [Figure 2: see original paper]). To ensure visually seamless results, patch selection methods and pixel writing algorithms must be carefully designed.

In the ATIPF algorithm, steps 1–2 constitute the "guidance map" generation process, involving linear interpolation calculations between image blocks with computation proportional to the number of pixels in the target texture. Steps 3–4 involve finding the closest matching patches to the reference image, which randomly selects a fixed number of patches from the source texture and computes Euclidean distances against reference patches—this computation is proportional to the number of patches in the reference image and thus to the pixel count of the target texture. Steps 4–5 involve writing patches to the target texture, where solving the Poisson equation with fixed pixel dimensions and fixed Jacobi iteration steps also yields computation proportional to patch count and target texture pixel count. Consequently, ATIPF' s total computation scales linearly with target texture pixel count, giving it O(N) complexity.

Furthermore, ATIPF' s computational workflow exhibits high patch-level independence, making it suitable for parallelization and offering potential for implementation on multi-core CPUs or GPUs.

2 High-Dimensional Image Block Interpolation

The texture synthesis technique addressed in this paper must preserve various-scale features of the original texture example, maintaining both macroscopic feature distribution and microscopic details. Zhou et al.' s [16] texture analysis method can assign scalar parameters to each scale feature of example textures to guide block selection toward feature-consistent regions. However, their approach requires complex nonlinear dimensionality reduction calculations that are extremely slow for high-resolution images, failing to meet our need for fast computation.

This section proposes an image interpolation technique that rapidly generates intermediate texture images to serve as guidance for patch selection. The concept extends traditional pixel-level interpolation to high-dimensional patch-level interpolation.

Image interpolation has been extensively studied, primarily based on pixel-level

strategies, with the classic approach being bilinear interpolation (as shown in Figure 4 [Figure 4: see original paper]). The idea involves inserting new points between original image pixels while maintaining consistency with neighboring pixel characteristics:

However, classical bilinear interpolation destroys the original image's DPI due to its pixel-level computation strategy, resulting in blurred interpolation outcomes. This section proposes a patch-level interpolation approach that operates in high-dimensional space while maintaining computational efficiency.

If we generalize from pixel points to pixel blocks by first splitting the source texture into regular patches and uniformly distributing them across the target texture (as shown in Figure 5 [Figure 5: see original paper]), the interpolation problem can still be expressed in the form of Equation (1).

Experimental results show that interpolation algorithms of the form in Equation (2) produce noticeable block artifacts due to edge information carried by image patches, which are easily detected by human vision and cannot be directly applied.

To address this, this section proposes a radial basis function-like formulation [25–27] using mirror mapping and transition functions to interpolate smooth, block-artifact-free texture images. For clarity, gaps in Figure 5 are categorized into three types based on their positional relationship with source texture blocks.

Let horizontal mirror mapping (left-right flip) be denoted as and vertical mirror mapping (up-down flip) as . The high-dimensional texture interpolation formula proposed in this section is:

Since the transition basis function smoothly connects at patch edges, the interpolation result from Equation (3) shows no obvious block artifacts, only unnatural mirroring effects. Considering that the goal of this interpolation algorithm is to provide guidance and criteria for patch searching, its effect fully meets texture interpolation requirements.

3 Poisson Filling

While the high-dimensional interpolation method proposed in Section 3 achieves texture scaling without block seams, its results contain numerous unnatural mirrored regions. For highly periodic complex textures, this mirroring effect is less noticeable, but for general non-periodic textures—especially non-stationary ones—it is easily detected by human vision. However, this intermediate result can guide the patch-finding algorithm to select patches from the source texture that best approximate the intermediate texture, which are then used to replace gap region pixels.

When replacing gaps, smooth transitions between gaps and source texture blocks must be preserved, requiring an appropriate patch blending method. Traditional texture synthesis algorithms have proposed many such methods with successful

results for stationary textures. To avoid time-consuming complex optimization or boundary optimization segmentation, this paper selects Poisson image editing [28] as the means to embed patches into intermediate texture gaps.

The principle of Poisson image editing is straightforward, with its core idea being to solve the Poisson equation over region :

where is the image block to be solved and is the Laplacian operator.

Let the texture patch found from the source texture that best matches the gap (using this gap type as an example; other gap types follow identical processes) be denoted as . Its Laplacian transform result and boundary conditions can be substituted into Equation (4) to establish the Poisson equation:

The optimal texture patch can be simply defined using pixel-wise squared distance:

where is any sub-region patch from the source example texture. In practice, this paper employs Monte Carlo sampling to randomly extract patch samples as the search space for Equation (6).

As shown in Figure 8 [Figure 8: see original paper], the texture image after Poisson filling has eliminated the mirroring effects of the intermediate texture while fully preserving its feature distribution, thereby maintaining the feature distribution of the source example texture. Furthermore, since all patches used in Poisson filling come directly from the source texture image without additional image processing, their DPI remains consistent with the source texture, achieving the goal of stretching the source texture without clarity loss.

4 Computational Results and Discussion

This section presents experimental results compared with classical algorithms to demonstrate that the proposed ATIPF algorithm can perform fast interpolation on both stationary and non-stationary texture images. All results were obtained on a PC platform with a 3.4 GHz Intel CPU, programmed in Matlab 2015b. Results from previous algorithms are those published by their authors. For clarity, abbreviations for referenced algorithms are listed in Table 1 .

Table 1 Algorithm abbreviations and corresponding literature numbers

Abbreviation	Literature
Image Quilting	[4]
Texture Optimization	[12]
Graphcut Textures	[13]
Image Melding	[14]
Self Tuning	[11]
DeepCor	[17]
NTSAE	[18]

4.1 Comparative Experiments on Stationary Texture Interpolation

For stationary textures, source images contain many repetitive texture features at the same scale, with synthesis results primarily involving random extraction of patch pixel information from the source texture followed by splicing into the target texture. To avoid obvious periodic repetition or seam artifacts, existing algorithms employ optimal boundary cutting (image quilting) [4] or optimal patch region selection (texture optimization) [12]. As shown in Figure 9 [Figure 9: see original paper], stationary texture synthesis is relatively straightforward, with classical algorithms generally producing seamless results effectively. The advantage of the proposed algorithm lies in preserving the source texture' s feature distribution pattern due to its use of high-dimensional image interpolation results as synthesis guidance. For instance, Figure 9c contains only a few large green leaf regions whose distribution positions are inconsistent with the source image.

4.2 Comparative Experiments on Non-Stationary Texture Interpolation

Non-stationary textures are generally natural photographic textures featuring multi-scale structural characteristics with non-periodic, irregular spatial distribution properties—making them unsuitable for classical texture synthesis algorithms. Additionally, natural photos typically have high resolutions, significantly expanding the search space for optimization-based algorithms and rapidly decreasing their computational efficiency.

Kaspar et al.' s [11] Self Tuning algorithm can adapt to weakly non-stationary texture synthesis scenarios, but its multi-scale optimization process is time-consuming and unsuitable for strongly non-stationary textures. Zhou et al. [16] proposed a "Guidance Map" approach to guide Self Tuning for non-stationary texture synthesis, achieving good results. However, Guidance Map computation involves nonlinear dimensionality reduction that is extremely slow for high-resolution source textures. This section presents comparison results between the proposed ATIPF algorithm and Self Tuning.

Figure 10 [Figure 10: see original paper] shows comparative synthesis results for simple non-stationary texture source images, where Figure 10b presents Image Melding [14] results, Figure 10c shows Graphcut Textures [13] results, Figure 10d displays Self Tuning [11] results, and Figure 10e illustrates the proposed ATIPF algorithm results.

Unoptimized texture synthesis algorithms tend to repeatedly select similar texture patches, resulting in overly flat outcomes (Figure 10(b)) or obvious background segmentation artifacts (Figure 10(c)). Self Tuning [11] achieves better results through complex global optimization that uniformly selects all patches from the source texture, but cannot maintain consistency with the source texture' s feature distribution (Figure 10(d)). The proposed ATIPF algorithm eliminates the need for global optimization, instead guiding patch selection through

texture interpolation to generate synthesized textures consistent with source texture feature distribution (Figure 10(e)).

Notably, since ATIPF employs Poisson blending to embed patches, it may introduce slight color "bleeding" artifacts at a small number of patch boundaries (as seen in the upper row of Figure 10(e)).

4.3 Experiments on Complex Texture Interpolation

For complex non-stationary textures, traditional patch-based algorithms lack descriptions of multi-scale features in source textures, making them unable to achieve satisfactory results even with global optimization [11]. To address this, Zhou et al. [16] proposed a "Guidance Map" strategy to guide patch searching, which improved results to some extent but remained unsuitable for high-resolution source textures due to its complex nonlinear dimensionality reduction. More recently, Zhou et al. [18] introduced a deep learning-based texture synthesis algorithm that significantly improved results for complex non-stationary textures.

Figure 11 [Figure 11: see original paper] shows the source texture containing two-scale features: nail-like protrusions on an iron plate oriented in two directions, with non-uniform brightness distribution caused by lighting. Self Tuning [11] adapts to the nail protrusion distribution through global optimization but alters the background brightness distribution (Figure 11(b)). DeepCor [17] fails to handle complex non-stationary textures, with its synthesis result (Figure 11(c)) essentially losing its texture synthesis function. NTSAE [18] successfully adapts to multi-scale texture feature non-uniformity, achieving the highest visual consistency. The proposed ATIPF algorithm shows slight overlap of nail protrusions but maintains background brightness distribution highly consistent with the source texture, outperforming even complex deep learning methods like NTSAE [18] in preserving source texture characteristics and demonstrating greater practical competitiveness.

Figure 12 [Figure 12: see original paper] presents interpolation results for high-resolution photographic textures (1024×600) using the ATIPF algorithm. For complex natural textures, the source texture' s inherent self-similarity can be fully exploited by the algorithm to produce interpolated textures with high visual consistency.

Although Zhou et al.' s [18] NTSAE learning algorithm has achieved surprising success on many complex textures (leaves, tree rings, peacock tails, etc.), its synthesis results always exhibit blurring at image boundaries. Figures 13 [Figure 13: see original paper] and 14 [Figure 14: see original paper] show two severe cases of this phenomenon, revealing the algorithm' s insufficient robustness. The proposed ATIFP algorithm demonstrates better robustness, effectively handling numerous complex texture interpolation problems by leveraging natural image self-similarity.

5 Conclusion

Based on high-dimensional image interpolation and Poisson filling, this paper proposes a texture interpolation algorithm that preserves image clarity while adaptively stretching source textures to target sizes. The algorithm avoids complex global optimization or deep learning training, requires low memory, and is easily parallelized, making it suitable for implementation on ordinary hardware platforms including mobile devices.

Compared with traditional texture synthesis algorithms, the proposed method demonstrates strong robustness, achieving good synthesis results for both stationary and non-stationary source textures. Additionally, the synthesis results maintain consistency with source texture feature distributions across all scales, making it highly suitable for natural texture image scaling and stretching applications.

Finally, for certain source textures with rotational feature repetition, the proposed algorithm' s synthesis quality is inferior to deep learning-based NTSAE methods, indicating a need for more appropriate patch searching schemes.

References

[1] Lagae A, Lefebvre S, Cook R, et al. A survey of procedural noise functions [J]. Computer Graphics Forum. 2010, 29 (8): 2579-2600.
[2] Barnes C, Zhang F. A survey of the state-of-the-art in patch-based synthesis [J]. Computational Visual Media. 2017, 3 (1): 3-20.
[3] Lasram A, Lefebvre S. Parallel patch-based texture synthesis [C]//Proc of the 12th Conference on High-Performance Graphics. Goslar: Eurographics Association, 2012: 115-124.
[4] Efros A A, Freeman W T. Image quilting for texture synthesis and transfer [C]//Proc of the 28th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press, 2001: 341-346.
[5] Efros A A, Leung T K. Texture synthesis by non-parametric sampling [C]//Proc of the 17th IEEE International Conference on Computer Vision. 1999: 1033-1038.
[6] Li Yang, Wu Minyuan, Yan Jia. Self-similarity based image super-resolution algorithm using optimized PatchMatch [J]. Application Research of Computers, 2018, 35 (4): 1231-1235.
[7] Shen Zhe, Wang Lili. Parallel algorithm for texture synthesis on GPU [J]. Journal of Computer-Aided Design & Computer Graphics, 2015, 27 (2): 330-336.
[8] Han JianWei, Wang Qing, Zhou Kun, et al. Wang tile based geometric texture synthesis [J]. Journal of Software, 2009, 20 (12): 3254-3264.
[9] Zou Kun, Han Guoqiang, Li Wen, et al. An efficient method of texture synthesis based on graph cuts [J]. Journal of Computer Aided Design & Computer Graphics, 2008, 20 (5): 652-658.
[10] Xu Xiaogang, Bao Hujun, Ma Lizhuang. Study on texture synthesis [J].

Journal of Computer Research and Development, 2002, 39 (11): 1405-1411.
[11] Kaspar A, Neubert B, Lischinski D, et al. Self-tuning texture optimization [J]. Computer Graphics Forum, 2015, 34 (2): 349-359.
[12] Kwatra V, Essa I, Bobick A, et al. Texture optimization for example-based synthesis [J]. ACM Trans on Graphics, 2005, 24 (3): 795-802.
[13] Kwatra V, Dl Sch O A, Essa I, et al. Graphcut textures: image and video synthesis using graph cuts [J]. ACM Trans on Graphics, 2003, 22 (3): 277-286.
[14] Darabi S, Shechtman E, Goldman D B, et al. Image melding: combining inconsistent images using patch-based synthesis [J]. ACM Trans on Graphics, 2012, 31 (4): 82.
[15] Lefebvre S, Hoppe H. Appearance-space texture synthesis [J]. ACM Trans on Graphics, 2006, 25 (3): 541-548.
[16] Zhou Y, Shi H, Lischinski D, et al. Analysis and controlled synthesis of inhomogeneous textures [J]. Computer Graphics Forum, 2017, 36 (2): 315-325.
[17] Sendik O, Cohen-Or D. Deep correlations for texture synthesis [J]. ACM Trans on Graphics, 2017, 36 (5): 161.
[18] Zhou Y, Zhu Z, Bai X, et al. Non-stationary texture synthesis by adversarial expansion [J]. ACM Trans on Graphics, 2018, 37 (4): 49.
[19] Yu Siquan, Han Zhi, Tang Yandong, et al. Texture synthesis method based on generative adversarial networks [J]. Infrared and Laser Engineering, 2018, 47 (2): 34-39.
[20] Lin S, Yeh I, Lin C, et al. Patch-based image warping for content-aware retargeting [J]. IEEE Trans on Multimedia, 2013, 15 (2): 359-368.
[21] Penfei Yue, Huayu Wang, Yuanjie Zhen, et al. Image retargeting using blur based depth saliency descriptor [J]. Journal of Computer Aided Design & Computer Graphics, 2018, 30 (3): 415-423.
[22] Gu Xiangli, Chi Jing, Zhang Caiming. Image scaling based on the spring deformation model [J]. Journal of Image and Graphics, 2018, 23 (5): 756-765.
[23] Cun Liu, Yuanxiang Li, Yongjun Zhou, et al. Video image super-resolution reconstruction method based on convolutional neural network [J/OL]. Application Research of Computers, 2019, 36 (4). http://www.arocmag.com/article/02-2019-04-057.html.
[24] Nie Dongdong, Ma Lizhuang. Fast image resizing method under energy balance principle [J]. Journal of Computer Aided Design & Computer Graphics, 2014, 26 (8): 1314-1319.
[25] Zhixiang C, Feilong C. Spherical scattered data quasi-interpolation by gaussian radial basis function [J]. Chinese Annals of Mathematics: Series B, 2015 (3): 401-412.
[26] Carr J C, Beatson R K, Cherrie J B, et al. Reconstruction and representation of 3D objects with radial basis functions [C]//Proc of the 28th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press, 2001: 67-76.
[27] Beatson R K, Light W A, Billings S. Fast solution of the radial basis function interpolation equations: domain decomposition methods [J]. SIAM Journal on Scientific Computing, 2000, 22 (5): 1717-1740.
[28] Rez P E, Gangnet M, Blake A. Poisson image editing [J]. ACM Trans on

Graphics, 2003, 22 (3): 313-318.

Note: Figure translations are in progress. See original paper for figures.