IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 12 October 2023, accepted 22 October 2023, date of publication 27 October 2023, date of current version 2 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3328171

RESEARCH ARTICLE

Natural Image Decay With a Decay Effects Generator

GUOQING HAO[1,2], **SATOSHI IIZUKA**[1], **KENSHO HARA**[2], **HIROKATSU KATAOKA**[2], **AND KAZUHIRO FUKUI**[1], **(Member, IEEE)**
[1]Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba 305-0085, Japan
[2]National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba 305-8560, Japan

Corresponding author: Guoqing Hao (hao_guoqing@cvlab.cs.tsukuba.ac.jp)

This work was supported in part by the Japan Science and Technology Agency (JST), Support for Pioneering Research Initiated by the Next Generation (SPRING) under Grant JPMJSP2124, in part by the JST, Precursory Research for Embryonic Science and Technology (PRESTO) under Grant JPMJPR21C1, and in part by the Japan Society for the Promotion of Science (JSPS), Grants-in-Aid for Scientific Research (KAKENHI) under Grant JP21H04908.

ABSTRACT We present a novel framework for simulating time-varying decay effects for natural images. Conventional methods assume the input image includes enough decay information and uses the color or texture information of the decayed regions to transfer its effect to the non-decayed regions. Unlike these approaches, our framework generates diverse patterns of decay effects by leveraging a decay effects generator network without referencing the decay features of the input image, which allows us to handle more general images with non-decayed objects. Our decay generator network is formed by a style-based generative adversarial network with an arbitrary-sized stationary texture generation mechanism that allows us to synthesize various sizes of decay textures. This arbitrary-sized stationary texture generation is necessary to synthesize photo-realistic decay effects since the appropriate resolutions of the decay textures depend on those of the target objects. We construct a novel decay texture image dataset that contains various types of decay texture images, such as mossy and rust, to train the decay generator network. We show that our framework is able to synthesize diverse decay effects on various non-decayed objects without using additional decayed object images.

INDEX TERMS Arbitrary-sized image generation, generative adversarial networks, image decay, image editing, image manipulation, image processing, image weathering, texture generation.

I. INTRODUCTION

Realistic image manipulation is a long-standing goal in the fields of computer vision and graphics. Many efforts have been made to create realistic images, such as modeling the real world or simulating natural phenomena. Decay is a ubiquitous natural phenomenon in real-world objects, where the appearance of an object is often dramatically changed over time. For example, a rock in a moist environment may grow lichen. Simulating such a phenomenon is not only beneficial to enhance the realism of virtual objects but also important to the reproduction of the real world. However, this simulation is a laborious endeavor performed by experts and requires a significant amount of time.

Existing image-based decay approaches [1], [2], [3], [4], also known as image-based weathering, assume the input image contains enough decay information and extracts the color or texture features of the decayed regions to transfer their effects to the non-decayed regions. To handle non-decayed images with these approaches, the user should provide additional reference images with sufficiently-decayed objects. However, obtaining a suitable reference image for the input image is not trivial since there are variations even in a single type of decay effect, e.g., lichens in different environments may vary in appearance or structure.

Recently, image-to-image translation approaches based on convolutional neural networks [5], [6] have advanced rapidly and have shown remarkable performance in various image processing tasks, such as image super-resolution [7], image enhancement [8], [9], and image completion [10]. Very

The associate editor coordinating the review of this manuscript and approving it for publication was Yizhang Jiang.

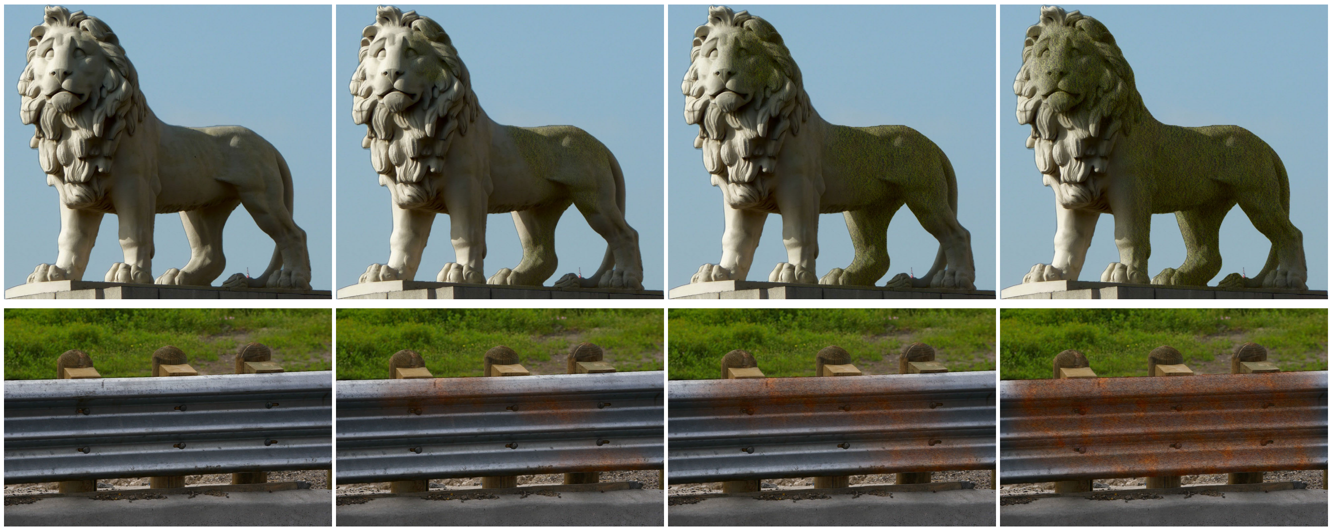


FIGURE 1. Time-varying decay results with the proposed method. Given a natural image (the first image in each row), our method is able to simulate the decay effects for non-decayed images without referencing additional decayed images, which is not possible with previous works.

recently, [11], [12] employed Diffusion Models (DM) [13] for the image-to-image translation approaches, achieving superior performance in various tasks and producing more realistic outputs than the methods of [5], [6]. However, applying the image-to-image translation approaches to the task of converting an input image into a decayed one is difficult. The supervised image-to-image translation method [5] requires a large number of paired training data, i.e., natural images and their corresponding decayed ones. In practice, however, it is impossible to construct such a large paired dataset as the decay process in the real world is a years-long or decades-long process. Although the unsupervised image-to-image translation method [6] does not need paired data for training, it only functions effectively on curated unpaired data that belong to analogous domains, e.g., decayed and non-decayed images with rough object shape matching or under similar scenes. Such curated unpaired data is also difficult to collect because of the lengthy decay process. More importantly, image-to-image translation methods cannot preserve details of the object shape that are not related to decay because they re-synthesize the objects.

Inspired by the work of Iizuka et al. [4], we bypass the need for paired data through a unique approach: generation and composition. We first generate a decay effect texture using a variant of generative adversarial networks (GANs) [14] and then synthesize them on the target object together with a global shading. Consequently, our natural image decay framework is able to simulate various time-varying decay effects in a single image without using external decayed images (FIGURE 1).

Our approach has several advantages. First, we can obtain diverse patterns of decay effects with the decay effect generator network, where the user can control the types of decay generation, such as rust and peeling. Second, we can easily preserve the details of the object shapes that are not related to decay in contrast to the image-to-image translation techniques that sometimes change them unintentionally. For example, image-to-image translation requires not only generating decayed regions but reproducing the remaining regions accurately through the translation network. In contrast, our method simply blends the decay effect texture and target regions with a global shading, which can stably preserve the appearance of the objects. Third, our framework does not require paired data or analogous data as training data for training the internal network. Our decay effects generator network is trained on unsupervised decayed texture images, which enables us to directly use texture images from the internet. We construct a novel decay texture dataset containing four types of decay texture images, i.e., rust, peeling, cracked, and mossy, which allows the generator to synthesize various types of decay effects.

In the decay effect generation, decay textures are required to be arbitrarily large while maintaining stationarity. Specifically, a decay texture containing repeating patterns with some randomness is necessary to be arbitrarily large since the appropriate resolutions of the decay textures depend on those of the target objects. A visualization of the importance of arbitrary-sized stationary texture is shown in FIGURE 2. We observe inconsistent sizes between decay texture and the target object cause structure artifacts. Besides, the result of InfinityGAN demonstrates that non-stationary texture produces a chaotic appearance. To address the problems, we introduce an arbitrary-sized stationary decay texture generation mechanism into the decay generator. This arbitrary-sized stationary texture generation allows for simulating photo-realistic decay effects for natural images.

We quantitatively evaluate the quality of generated decay texture against other generative texture models in terms of Fréchet Inception Distance (FID) [17]. Additionally, we propose to use the Gram matrix [18], [19], [20] to

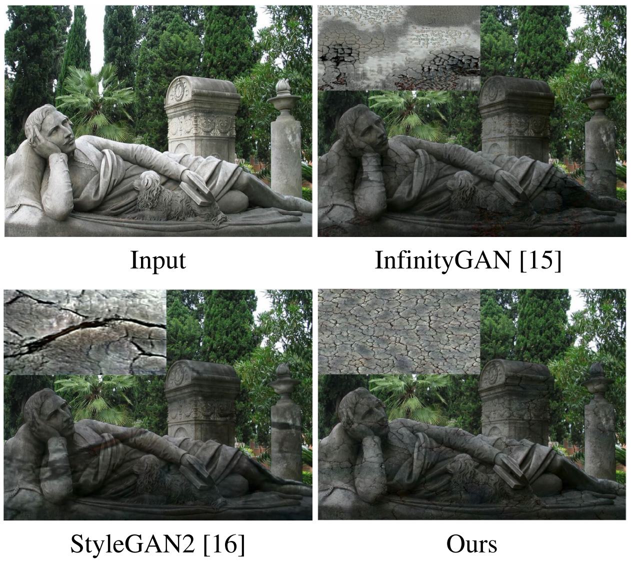


FIGURE 2. The importance of arbitrary-sized stationary texture in the proposed framework. The result of the StyleGAN2 is synthesized by resizing a 128 × 128 pixels texture, while other results are generated by matching the size of the output with that of the input image. The InfinityGAN [15] generates arbitrary-sized non-stationary textures, whereas our method generates arbitrary-sized stationary textures.

measure the style similarity of the texture patches, which allows us to evaluate the spatial stationarity of the generated decay texture. We also perform a perceptual user study on decayed results of a broad range of natural images, where our framework shows a significant performance over other generative texture models [15].

Our contributions are as follows:

- We propose a novel framework that can generate diverse decay effects on non-decayed images without referencing other decayed images.
- Our approach incorporates a decay effect generation mechanism capable of generating stationary and arbitrary-sized decay textures. This allows for a realistic simulation of decay effects on natural images.
- We construct a unique decay texture dataset that includes major categories of real-world decay, ensuring the diversity of decay effects.

II. RELATED WORK

In this section, we discuss the related work on decay simulation, texture synthesis, and arbitrary-sized image generation.

A. DECAY SIMULATION

Simulating time-varying decay effects has been extensively explored over the past three decades. We discuss the related work that addresses the problem of decay simulation.

1) PHYSICALLY-BASED DECAY

Several approaches [23], [24], [25], [26], [27], [28], [29], [30] generate decay effects using physically-based simulation. They have achieved accurate results by addressing the simulation of specific decay effects, such as metallic patinas [23], stone erosion [25], paint cracks and peeling [26], water flow [24], corrosion [31], stretches [27], and lichen growth [28]. Although this type of method produces reliable decay results, it cannot generalize well to pending modeling scenarios. A set of techniques [32], [33], [34] creates convincing decay results for different decay effects by capturing the time-varying reflectance of materials. However, these approaches require an understanding of geometrical details and the surrounding environment of an object, which is difficult to obtain from a natural image.

2) IMAGE-BASED DECAY

To model more general decay effects of images, Wang et al. [35] introduced appearance manifolds, which model the time-variant surface appearance of material from data captured at a single instant in time. Although their method mainly targets 3D models, they also made a simple attempt to model decay effects from a single image. Xue et al. [1] thoroughly extended the appearance manifolds method to the single image decay scenario. They developed an iterative method to construct the appearance manifold in color space for modeling the reflectance variations. Bandeira and Walter [2] simplified the appearance manifold to a two-dimensional appearance map, which significantly reduces computational costs. Endo et al. [3] further improved this method by extracting fine-scale geometries as high-frequency components of the image. Iizuka et al. [4] proposed a texture variations-based method to model and propagate complicated appearance variations. Specifically, they first model the distribution of decay degrees using Radial Basis Functions. Afterward, they propagate the appearance features by sampling patches from decayed regions and compositing them on a target surface according to the decay degree. Most recently, Du et al. [36] introduced a new image-based decay technique that automatically extracts multiple exemplars from different decay stages, giving the user the flexibility to suit different applications, such as customized decay generation and decay transfer.

Nonetheless, those existing methods cannot simulate decay effects on non-decayed objects without referencing the decay features (e.g., color or texture) on decayed regions. To simulate decay effects on non-decayed objects with previous work, a reference decayed image is collected and analyzed to transfer its decay effects to non-decayed objects. However, it is non-trivial to obtain an appropriate reference decayed image because there exist large variations even in a single type of decay effect. Thus, natural image decay still remains a challenging task.

Unlike previous work [1], [2], [3], [4] exclusively relies on the use of decayed objects, we generate various decay effects with a decay effects generator to process more general images without referencing the decayed regions. In particular, our framework adopts the strategy of [4] that simulates decayed

TABLE 1. **Comparison of different approaches for simulating decay effects. Patch-based [21] and learning-based [22] texture decay methods can only be applied to the texture space. Conventional image-based decay approaches, including low-level feature variation [1], [2], [3] and texture variation [4], simulate decay effects by transferring decayed features. More importantly, none of the existing methods can produce diverse patterns of decay effects. In contrast, our method can generate diverse patterns of decay results for arbitrary-sized natural images.**

	Target image	Approach	Diverse patterns	Image size
Patch-based texture decay [21]	Texture	Transfer	No	Any
Learning-based texture decay [22]	Texture	Generation	No	Fixed
Low-level feature variation [1]–[3]	Natural image	Transfer	No	Any
Texture variation [4]	Natural image	Transfer	No	Any
Ours	Natural image	Generation	Yes	Any

results by compositing decay texture onto the target surface according to a decay degree.

The decay effects in objects can be decomposed into two aspects: appearance and geometry changes. Traditional image-based decay methods [1], [2], [4] focus on appearance changes while ignoring the geometry changes. To achieve geometry-aware decay, we considered using the normal map to model geometry details, which requires accurate normal estimations from images. However, we found existing normal estimation methods generalize poorly on natural images. In contrast, we observed the shading-aware strategy used in previous work [4] produces accurate results on most natural images. Therefore, we employ a robust shading-aware strategy, following the strategy used in [4].

3) TEXTURE DECAY

Recently, a problem of texture space decay has been addressed. Bellini et al. [21] presented a technique to synthesize time-varying decayed textures. Specifically, given a single decayed input texture, they estimate the decay degree at different regions by prevalence analysis of texture patches. With the estimated decay degree, they can generate backward and forward time series to create decay and inverse decay effects. Chen et al. [22] leveraged this method as a basis and proposed to generate the decay process using the image-to-image translation technique [5]. They first analyzed the input texture to obtain paired data of the decay degree and an input image. Then, an image-to-image translation model is trained to generate texture patches for a single input. After training, new decay results can be generated by modifying the decay degree. Although their work [21], [22] can only be applied to the texture space, one might imagine generating a decayed image from a natural image with an image-to-image translation model [5], [12]. In practice, however, we cannot employ the image-to-image translation method directly in the task of natural image decay because it is impossible to collect a sufficient number of training paired data of natural images and their decayed ones.

Our framework overcomes the limitations of the previous works by generating decay textures with a deep generative model, not propagating existing weathering features of weathered regions. A high-level comparison of different methods is shown in TABLE 1. Conventional image-based decay approaches, including low-level feature variation methods [1], [2], [3], texture variation [4], and patch-based texture decay [21], can only transfer decay effects, which rely on the use of the decayed feature. More importantly, none of the existing methods can produce diverse patterns of decay effects. In contrast, our method can generate diverse patterns of decay results for arbitrary-sized natural images.

B. TEXTURE SYNTHESIS

Most existing texture synthesis methods [37], [38], [39], [40], [41] generate textures from a given exemplar. Traditional texture synthesis methods use non-parametric or optimization-based techniques to synthesize a texture image while considering the exemplar. Therefore, none of the traditional texture synthesis methods can generate textures from scratch. Recently, a group of works [42], [43] has been proposed to generate textures from scratch with a generative adversarial network. However, their work can only generate small-size realistic textures and shows less diversity in generated textures.

C. ARBITRARY-SIZED IMAGE GENERATION

Generative Adversarial Networks (GANs) [14] have shown a remarkable ability in fixed-size image synthesis. Although several approaches [16], [44], [45] have been proposed to address the problem of high-resolution image synthesis, conventional GANs cannot generate arbitrary-sized images because of the limitation of computational resources. Several works [15], [46], [47], [48] have been proposed to address the problem of arbitrary-resolution image generation. Lin et al. [48] generated large images in a seamless manner by learning a coordinate manifold. Although Lin et al. [48] can generate seamless high-resolution images, they can only extrapolate a few patches beyond the training patch size. Lin et al. [15] further introduced a padding-free generator and a coordinate-based implicit function to generate arbitrary-sized images in a seamless synthesis-by-part manner. This method sets a state-of-the-art in landscape image generation.

Recently, a few attempts have been made at the task of generative texture synthesis. We refer to texture as

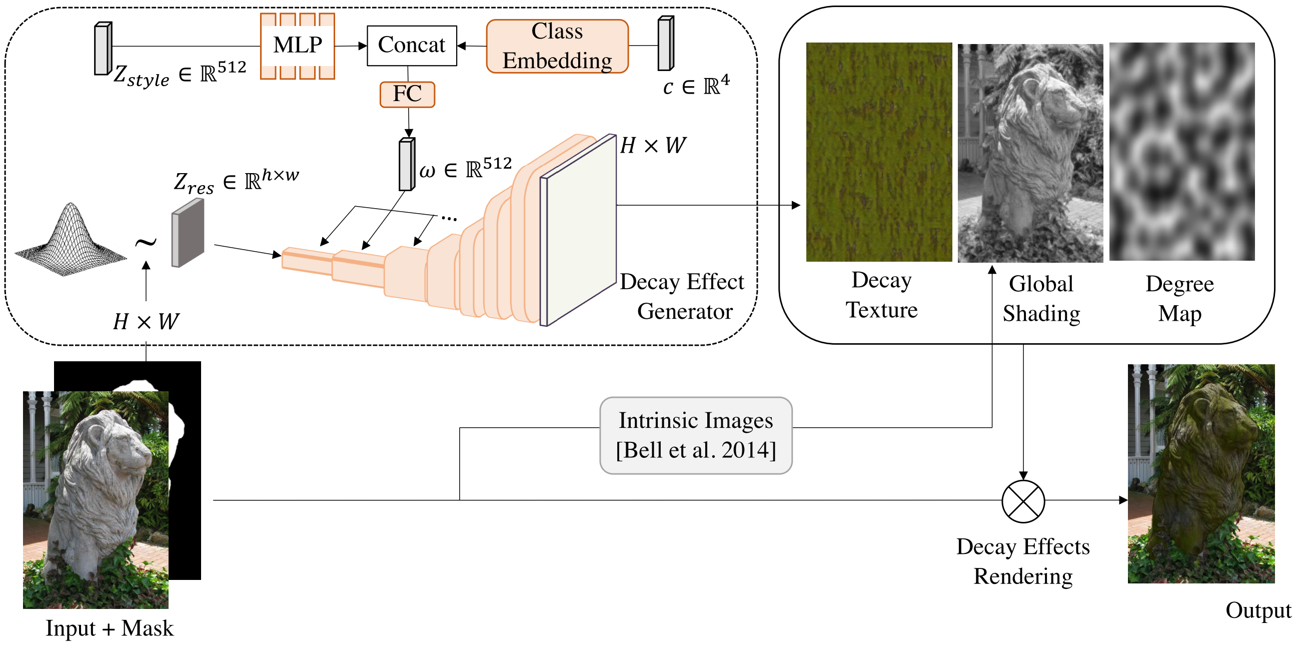


FIGURE 3. Overview of the proposed framework. Given an input image and a manually labeled corresponding mask, the decay effect generator takes three sets of inputs, i.e., a user-specified one-hot class vector **c**, a style code $\mathbf{Z_{style}}$ that controls the overall style of the output texture, and a stationary flexible-sized input $\mathbf{Z_{res}}$ that affects the resolution of the output texture. Afterward, the generator outputs a decay effect texture that is used for the decay effects rendering, where the final output image is synthesized with the input image, decay texture, and global shading, according to the degree map.

the terminology in texture synthesis [37], where textures are images containing repeating patterns with some randomnesses(spatial stationarity). Li et al. [49] presented a deep generative network for generating diverse outputs of multiple types of textures. Jetchev et al. [42] modified the DCGAN [50] architecture to allow for scalability and the ability to create any desired output texture size. They [43] further improved the [42] by extending the structure of the input noise distribution to gain a better learning capacity. They also showed the improved version has a property of texture manifolds, where smooth interpolation between generated samples is completed. Although arbitrary-sized texture synthesis [42], [43] has shown impressive empirical results, a theoretical basis is absent in this task. Lu et al. [51] presented a comprehensive theoretical interpretation for arbitrary-sized texture synthesis. They showed that marginalization consistency and permutation invariance are fundamental to synthesis-by-part texture synthesis. To satisfy these two properties, they extensively analyzed convolutional neural networks (CNNs) building blocks and provided instructions on modification of the current CNNs building blocks. Although Lu et al. [51] provided a comprehensive interpretation for arbitrary-sized texture synthesis, their proposed framework showed a learning capacity shortage due to the use of simple GANs architecture. Our approach is related to the work of [51], we propose a decay effects generation, which is based on the StyleGAN2 [16] with an arbitrary-sized stationary generation mechanism, to generate a spatial stationary decay texture.

TABLE 2. A specification shows the number of images of each category and the overall dataset.

Category	Rust	Peeling	Cracked	Mossy	Total
Number	1,173	964	496	735	3,368

III. APPROACH

An overview of our framework is shown in FIGURE 3. Following the previous works of natural image weathering [1], [2], [3], [4], the input is a natural image with a corresponding mask that indicates the region of a target object. We first generate a decay texture with the decay effects generator. The decay effects generator takes three inputs, a class **c** is specified by users, a style code $\mathbf{Z_{style}}$ is randomly sampled to represent the overall style of generated texture, and a white Gaussian noise $\mathbf{Z_{res}}$ of which size is modified to generate a texture that matches the size of the target object. After that, we calculate a global shading that describes the surface details of the object with an image intrinsic decomposition method [52]. And then, we initialize a degree map that represents the decay degree of each pixel. The degree map is initialized by sampling random patterns inside object regions and filling them with the Perlin noise. Finally, we follow the rendering step in [4] to simulate time-varying effects with the decay texture, a global shading, a degree map, and the input image.

A. DIVERSE TEXTURE SYNTHESIS

To generate various decay effects, we create a novel decay texture dataset that includes four categories of decay textures, i.e., rust, mossy, peeling, and cracked. In detail, we first crawl texture images from Google and Naver websites by using names of different decay textures as keywords to increase the diversity of the dataset. Although we obtain a large number of different textures, not all of these are qualified to train the decay effects generator. Most collected textures are extremely noisy, such as including a watermark or an easily recognizable object. Therefore, we apply a manual filtering process to remove noisy images that are unqualified to train the decay effects generator. In total, the decay texture dataset contains 3,368 images that are divided into 4 categories of decay textures, including rust, cracked, mossy, and peeling textures. A specification of the dataset is shown in TABLE 2. Several examples are shown in FIGURE 4.

Furthermore, we apply a mode-seeking diversity loss [53] that encourages the generators to be able to generate diverse patterns of decay textures. This mode-seeking diversity loss is described as follows:

$$L_{ms} = \frac{\left\| G(\mathbf{c}, \mathbf{Z}^{1}_{\mathbf{style}}) - G(\mathbf{c}, \mathbf{Z}^{2}_{\mathbf{style}}) \right\|_{1}}{\left\| \mathbf{Z}^{1}_{\mathbf{style}} - \mathbf{Z}^{2}_{\mathbf{style}} \right\|_{1}}, \tag{1}$$

where $\mathbf{Z}^{1}_{\mathbf{style}}$ and $\mathbf{Z}^{2}_{\mathbf{style}}$ are different style inputs, G is the decay effects generator, and $\mathbf{c}$ is the class label.

B. ARBITRARY-SIZED DECAY TEXTURE GENERATION

Our framework generates arbitrary-sized stationary decay textures, matching the sizes of the texture with those of the input image, to produce photo-realistic decay effects. We achieve seamless arbitrary-sized textures generation with a decay effects generator that is based on a StyelGAN2 [16] model. In particular, the decay effects generator takes three inputs: a user-specified class label $\mathbf{c}$, a randomly sampled style code $\mathbf{Z_{style}}$, and a spatial stationary Gaussian noise $\mathbf{Z_{res}}$. We can generate diverse patterns of textures in varying sizes by manipulating inputs. The generation process can be formally described as follows:

$$\begin{aligned} \omega &= FC(Concat(MLP(\mathbf{Z_{style}}), CE(\mathbf{c}))) \\ \mathbf{T}^{H\times W} &= G(\omega, \mathbf{Z_{res}}^{h\times w}). \end{aligned} \tag{2}$$

Here, *MLP* and *CE* are both embedding functions that are based on a Multilayer Perceptron. The *MLP* maps a $\mathbf{Z_{style}} \in \mathbb{R}^{512}$ to an intermediate latent space. The class embedding *CE* layer maps a one-hot class vector $\mathbf{c} \in \mathbb{R}^{4}$ to the same dimensions as the latent $\mathbf{Z_{style}}$. *Concat* operation is conducted in the channel dimension. The *FC* represents a fully connected layer that maps $\omega \in \mathbb{R}^{1024}$ to $\omega \in \mathbb{R}^{512}$. After

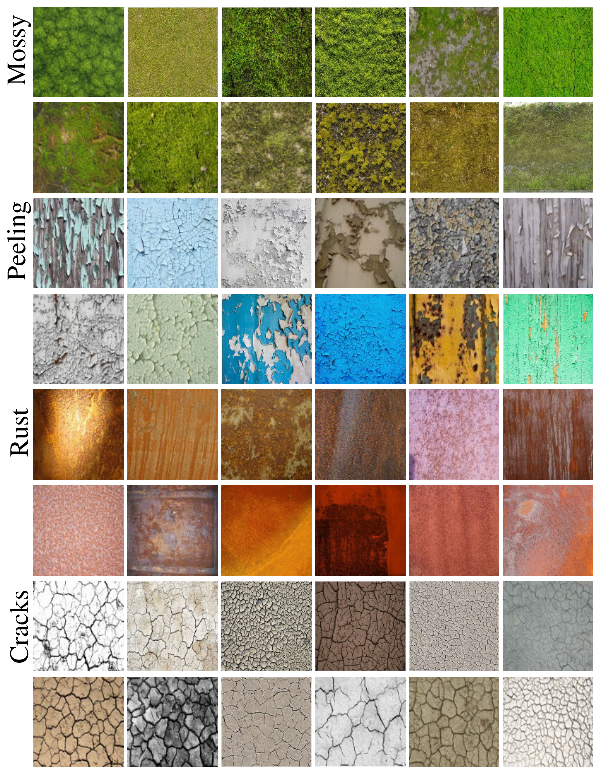


FIGURE 4. Examples of our decay texture dataset. We collect a novel decay texture dataset that includes four categories of decay textures, i.e., rust, mossy, peeling, and cracked.

that, a style code ω is fed into the decay effects generator G with a spatial stationary Gaussian noise $\mathbf{Z_{res}} \in \mathbb{R}^{h\times w}$ to generate a decay texture $\mathbf{T} \in \mathbb{R}^{H\times W}$. The $h \times w$ is set to 12×12 to generate a 132×132 image in the training phase, while can be arbitrarily set in the inference phase.

We introduce an arbitrary-sized stationary texture generation mechanism into the decay effects generator. Specifically, we replace a learned constant input of the StyleGAN2 with a flexible-sized spatial input to generate arbitrarily large textures. A decay texture, formally described as a realization of a stationary ergodic stochastic process [54], is generated by sampling the flexible-sized input from white Gaussian noise. The white Gaussian noise can also be formulated as a stationary ergodic stochastic process. This stationary input provides stationarity to the decay effects generator to generate stationary decay textures. However, conventional GANs do not transform a stationary input into another stationary output due to the abuse of inconsistent transformation. For example, the zero-padding layer in convolutional neural networks (CNNs) is a typical inconsistent transformation, which induces redundant positional information and produces noticeable inconsistent structure artifacts in the output.

Drawing inspiration from Lu et al.'s work [51], we meticulously redesign the architecture of the StyleGAN2 to form consistent transformations, which preserves stationarity between input and output. In detail, we first modify all convolution layers with zero-padding to padding-free convolution layers. And then, all bilinear up-sampling layers are followed by a boundary crop operation where boundary pixels are discarded. By introducing an arbitrary-sized stationary texture generation mechanism, the decay effects generator can generate arbitrary-sized stationary textures in a seamless manner. We encourage those interested in a deeper dive into the theoretical underpinnings of our modifications to refer to the foundational paper by Lu et al. [51].

C. DECAY EFFECTS RENDERING

1) GLOBAL SHADING

Due to the fact that the visual appearance of an object is replaced with the decay texture in a decay process, geometrical details of the object are easily lost. Following the previous work [4], we use a global shading that describes the geometrical details of an object to preserve its geometrical details in the decay process. To compute the global shading, we leverage an intrinsic image algorithm [52] based on a dense conditional random field (CRF) formulation [55] that considers long-range material relations.

2) DECAY DEGREE INITIALIZATION

In a decay process, we utilize a decay degree to control decay effects. A range of a decay degree spans from 0 to 1, in which 0 indicates that the object retains its original appearance, while 1 represents that the appearance of an object is completely replaced with the decay texture. To define a degree, we first generate random patterns in the regions of the object, where continuity and density can be controlled by sampling different parameters. We then fill the random patterns with the Perlin noise commonly used to increase the realism of synthetic textures in computer graphics. A visualization of synthetic degrees is shown in FIGURE 5.

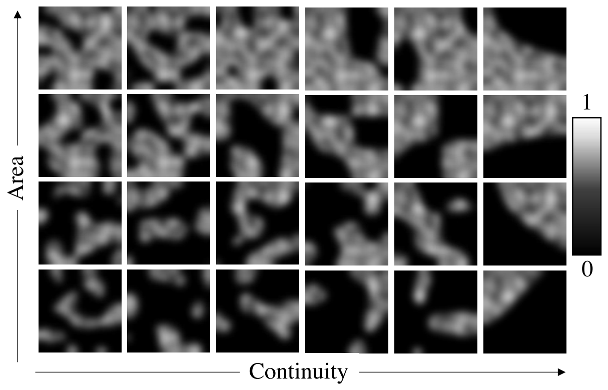


FIGURE 5. Decay degree initialization. We use random patterns filled with the Perlin noise to initialize a degree that controls the decay effects. The random pattern generation can be controlled by two parameters: continuity and area.

3) DECAY SYNTHESIS

To render the final output for a given input image, we employ a decay effect rendering step to generate the final output $\mathbf{O}$ with the decay texture $\mathbf{T}$, a degree $\mathbf{D}$, a global shading $\mathbf{S}$, the input image $\mathbf{I}$, and the corresponding mask $\mathbf{M}$. In detail, we first convert an input image and a generated decay texture from the RGB space into the LAB space. And then, the luminance channel of the input image is adjusted with the luminance channel of the decay texture according to the degree. This process can be formally described as:

$$\begin{aligned}\mathbf{O}_{\mathbf{lum}} &= (\mathbf{D}\mathbf{T}_{\mathbf{lum}} + (\mathbf{1}-\mathbf{D})\mathbf{I}_{\mathbf{hue}}) \odot \mathbf{S},\\ \mathbf{O}_{\mathbf{hue}} &= \mathbf{D}\mathbf{T}_{\mathbf{hue}} + (\mathbf{1}-\mathbf{D})\mathbf{I}_{\mathbf{hue}},\end{aligned} \tag{3}$$

where $\mathbf{1}$ represents an all-one matrix with the same shape of $\mathbf{D}$, $\odot$ denotes Hadamard product, and the terms with subscripts lum and hue represent the luminance channel and chroma channels in Lab space, respectively.

To further enhance the realism, we apply a color transfer algorithm [56] to match the color distribution between the decay texture and input image for peeling and cracked effects before the rendering step. Armed with our proposed framework, it is possible to simulate diverse patterns of decay effects for non-decayed objects.

4) DECAY EFFECTS DIFFUSION

We introduce a diffusion concept to the time-varying simulation of decay effects. Diffusion is the movement of molecules from a region of higher concentration to a region of lower concentration down the concentration gradient. We assume that the time-varying decay process fulfills the diffusion rule along with two regularizations: self-growth and non-decreasing. This is based on an observation that the natural decay process spreads out from highly-weathered regions and grows irreversibly in decayed regions. We compute the decay degree by solving a partial differential equation:

$$\frac{\partial \mathbf{D}}{\partial t} = \alpha\left(\frac{\partial^2 \mathbf{D}}{\partial x^2} + \frac{\partial^2 \mathbf{D}}{\partial y^2}\right), \tag{4}$$

where $\mathbf{D}$ is the decay degree to be solved, x and y are spatial coordinates, and t is time. The coefficient α is the diffusion coefficient that determines how fast $\mathbf{D}$ changes over time. The decay degree at time step $t+1$ can be computed by applying finite difference approximations and the decay regularizations:

$$\mathbf{D}^{t+1}_{x,y} = \mathbf{D}^{t}_{x,y} + \alpha \Delta t \times \mathbf{Div}^{t}_{x,y}, \tag{5}$$

where a finite difference approximation determines Δt, $\mathbf{Div}$ is the divergence at location x, y. And then, we apply two regularizations on $\mathbf{D}^{t+1}_{x,y}$:

$$\mathbf{D}^{t+1}_{x,y} = \begin{cases} \mathbf{D}^{t}_{x,y} + \mathbf{Z}_{x,y} & \text{if } \mathbf{D}^{t}_{x,y} > 0.25 \\ \mathbf{D}^{t}_{x,y} & \text{otherwise} \end{cases}, \tag{6}$$

$$\mathbf{D}^{t+1}_{x,y} = \mathit{Maximum}(\mathbf{D}^{t+1}_{x,y}, \mathbf{D}^{t}_{x,y}), \tag{7}$$

where $\mathbf{Z}_{x,y}$ is sampled from a uniform distribution to express self-growth property. Time-varying decay results can be found in Section C.

5) DECAY DEGREE EDITING

We also provide spatial control of the degree map to the users for interactive editing of decay effects. Specifically, we can initialize the degree map with user inputs to offer early-stage controls to the users, where users specify the beginning area of decay effects. We can also introduce spatial disparity to the decay effects diffusion by using spatial regularization, where the degree of a region is always higher/lower than the degree of other regions. This is especially important in editing objects with complicated geometry because some regions in objects with complicated geometry are easy or difficult to grow decay effects. As for the forms of user inputs, we can take as input varying forms, such as brush or control points.

IV. EXPERIMENTAL RESULTS

In this section, we discuss experimental details and demonstrate the decay results of different approaches.

A. EXPERIMENTAL DETAILS

1) MODEL TRAINING

The decay effects generator is trained on our decay texture dataset with a batch size of 16 for 900,000 iterations. The patch size used in the training phase is set to 132 × 132 pixels. For model training, we add the mode-seeking diversity loss [53] to the objective functions used in StyleGAN2 [16]. The overall loss function used to train the decay effects generator is described as follows:

$$L_{total} = L_{stylegan} + \lambda_{ms} L_{ms}, \tag{8}$$

where the weight λ_{ms} is set to be 1 during training, $L_{stylegan}$ and L_{ms} denote the original objective function and the mode-seeking diversity regularization, respectively.

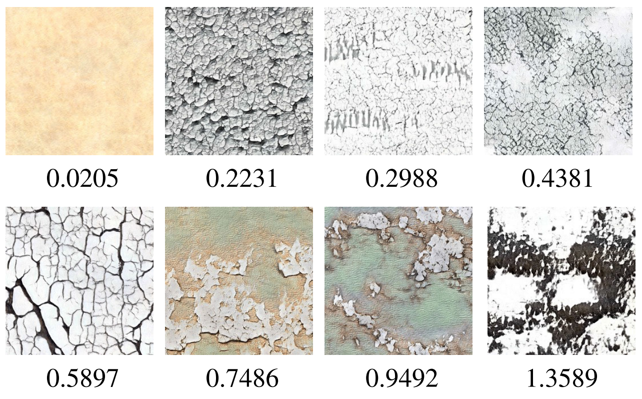


FIGURE 6. A visualization of texture images with different stationarities. Texture images with a lower metric of stationarity show a better perceptually local consistency. This supports the effectiveness of using style similarity between texture patches to assess stationarity. The metric of stationarity of each image is shown under the image.

2) COMPARISON WITH EXISTING APPROACHES

We compare our decay effects generator with several existing approaches in texture generation.

PSGAN: The PSGAN is a DCGAN-based [50] texture generator that allows for scalability and the ability to generate any desired output size. We use an official implementation to train the model and to generate arbitrary-sized textures.

CT-PSGAN: The CT-PSGAN is a modified version of the PSGAN based on the consistent transformation of the [51]. Although Lu et al. [51] provide a comprehensive interpretation of arbitrary-sized generation, their proposed method does not have enough learning capacity to generate diverse decay textures. Therefore, we modify the PSGAN architecture, making it satisfy the consistent transformation proposed by [51], to compare with our proposed model.

InfinityGAN: We also compare our decay effects generator with InfinityGAN [15], which is the state-of-the-art arbitrary-sized image generation approach. We use an official implementation of InfinityGAN for training and inference.

For a fair comparison, we modified the baselines to conditional ones by giving a one-hot class vector. All approaches are trained using exactly the same training data. We use the officially released implementation if available.

3) METRICS

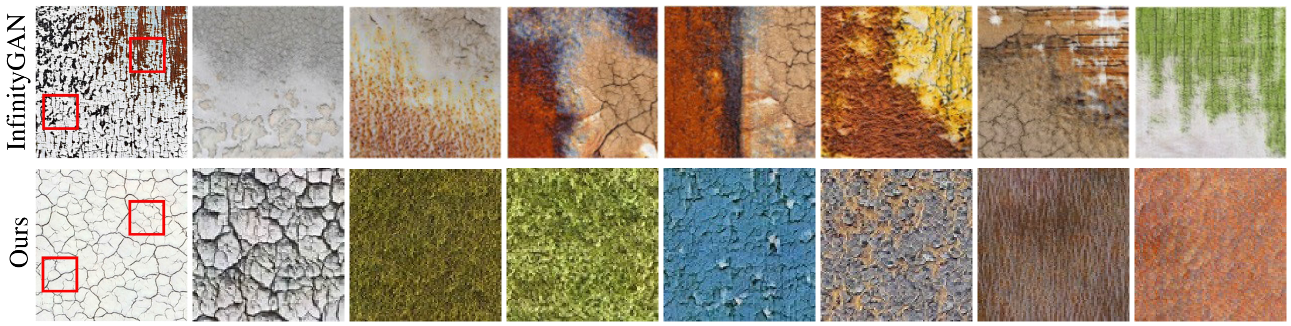
FID: To evaluate the quality of generated textures, we quantitatively compare our decay effects generator against other generative models in terms of Fréchet Inception Distance (FID) on different sizes of decay textures. Following the InfinityGAN, we resize all images into a fixed size before computing the FID.

Stationarity: Although the FID metric can evaluate the quality of generated textures, it cannot assess the stationarity of the generated textures. A stationary decay texture should contain repeating patterns with some randomness. In other words, the style of a texture patch is consistent with the styles of other texture patches in the same texture image. Based on the observation, we propose to use the Gram matrix [18], [19], [20] to assess the stationarity of textures. The Gram matrix represents the style of an input image with feature maps of the VGG16 network [57]. Suppose x with shape of $H_j \times W_j \times C_j$ is an input texture, the Gram matrix of the x is defined as:

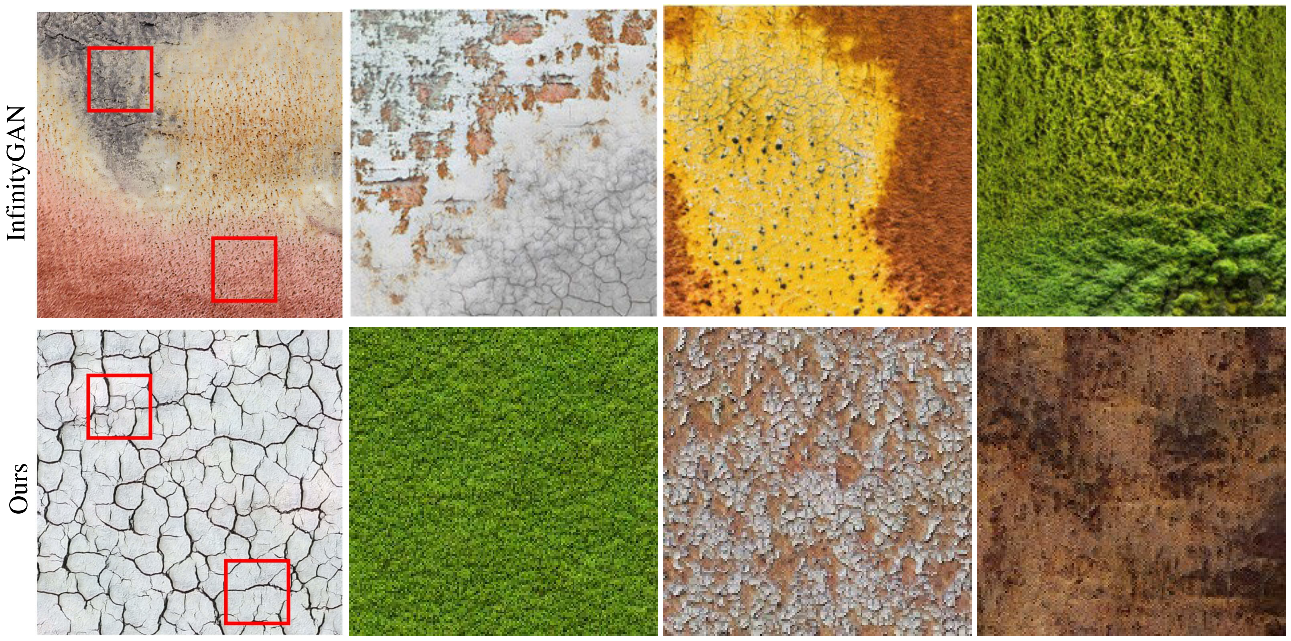
$$Gram_j(x) = \frac{1}{H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}, \tag{9}$$

where $\phi_j(x)$ is the activation at jth layer of the VGG16 network for the input x. The overall representation of the $Gram(x)$ is the sum of $Gram_j(x)$ for different layer j in the VGG16 network. Following the [20], we use the *relu1_2*, *relu2_2*, *relu3_3*, and *relu4_3* layers of the VGG16 network to calculate the overall Gram matrix.

Upon defining the Gram matrix, we measure the distance between the Gram matrices of different texture patches in the same texture image. Specifically, given a texture image



(a) Decay texture generation results at 256 × 256 pixels.



(b) Decay texture generation results at 512 × 512 pixels.

FIGURE 7. Comparative texture generation results using different approaches. Our method generates stationary decay texture, while InfinityGAN [15] generates locally varying textures. A stationary decay texture contains repeating patterns with some randomness. In other words, random texture patches are homogenous, as indicated by red boxes.

$\mathbf{T} \in \mathbb{R}^{H\times W\times C}$, we crop a set of random patches $\{(\mathbf{P_i}, \mathbf{Q_i})\}_{i=1}^{K}$ from it, where $\mathbf{P_i} \in \mathbb{R}^{h\times w\times C}$ and $\mathbf{Q_i} \in \mathbb{R}^{h\times w\times C}$ are non-overlapping patches, $h = H/N, w = W/N$, and K is the comparison time of the distance of Gram matrix between non-overlapping patches in the same texture image. The N is a constant that defines the scale of cropped patches. After that, we calculate the distances of the Gram matrix between non-overlapping patches K times and report the average value of those distances. The calculation of the stationarity d_{stat} is formally described as:

$$d_{stat} = \frac{1}{K}\sum_{i=1}^{K} \|Gram(\mathbf{P}_i) - Gram(\mathbf{Q}_i)\|_2 . \tag{10}$$

In our experiment, we set the N to be 4 and the K to be 10. For a better understanding of the stationarity metric, we provide several examples with different stationaries in FIGURE 6.

4) COMPUTATIONAL TIME

We measure the computational time of our method. Our decay effects generator takes 34ms to generate a 132 × 132 texture with a single Nvidia GTX 1080Ti GPU. More importantly, our method can generate larger textures with negligible additional time in a seamless synthesis-by-patch manner.

B. QUALITATIVE RESULTS

1) TEXTURE GENERATION

FIGURE 7(a) and FIGURE 7(b) demonstrate texture generation results of different approaches on varying sizes. We omit results of the PSGAN [43] and the CT-PSGAN [43], [51] in visualization as their methods generate unreasonable textures. From FIGURE 7(a) and FIGURE 7(b), we observe that InfinityGAN [15] does not generate stationary decay textures since their structure synthesizer tends to produce

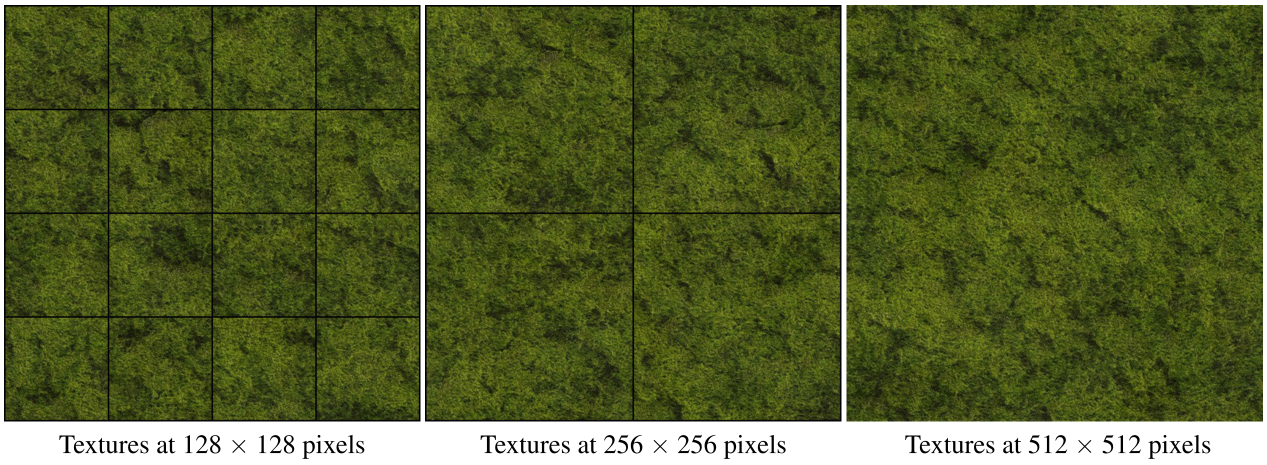

Textures at 128 × 128 pixels
Textures at 256 × 256 pixels
Textures at 512 × 512 pixels

FIGURE 8. Disentanglement results. We show decay textures by switching different resolutions of $\mathbf{Z}_{\mathbf{res}}$ while maintaining the same $\mathbf{Z}_{\mathbf{style}}$. The texture results demonstrate the $\mathbf{Z}_{\mathbf{style}}$ and the $\mathbf{Z}_{\mathbf{res}}$ are properly disentangled.

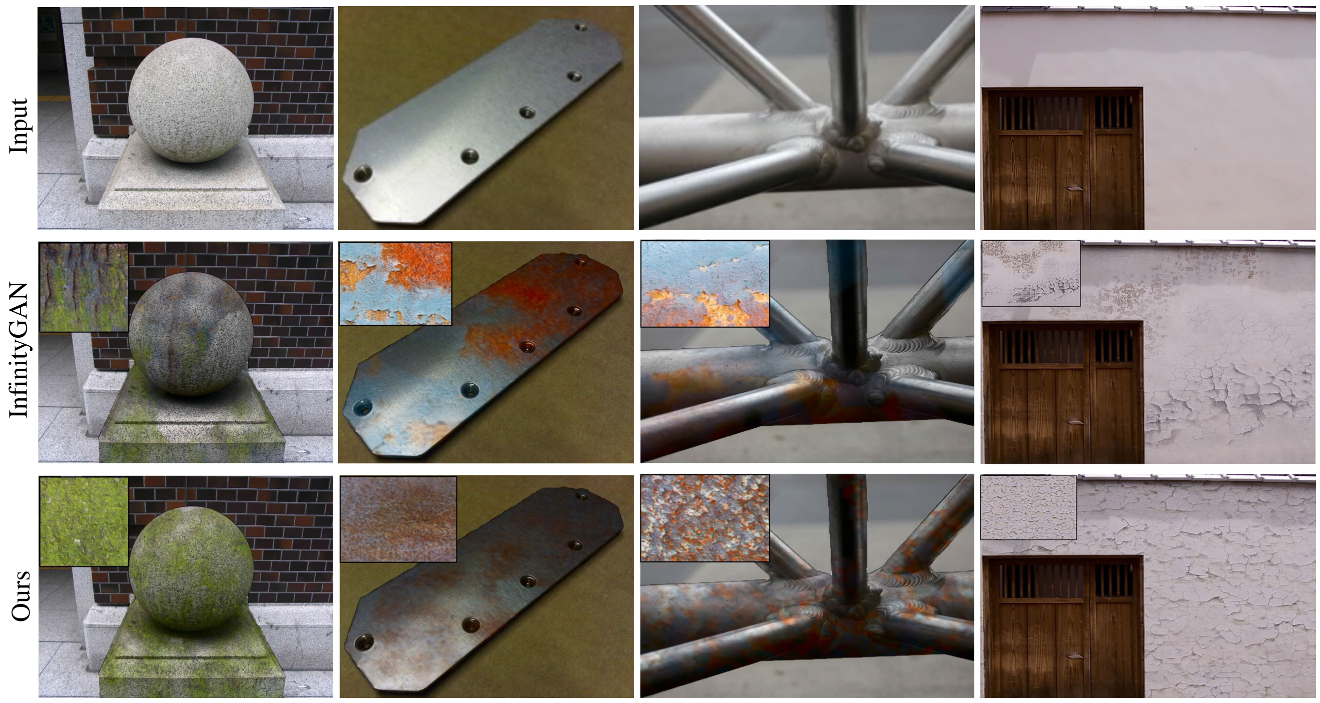


FIGURE 9. Comparisons with other generative texture models. We show decay results on different images and compare our approach with the InfinityGAN [15]. The first row shows the input image, and the last two rows are decay results generated by using different approaches. Textures used for rendering are put in the upper-left corner.

local structural variations in the output. In contrast, our method generates reasonable decay textures while maintaining stationarity with the arbitrary-sized stationary texture generation mechanism. The arbitrary-sized stationary texture generation mechanism consistently transforms a stationary flexible-sized Gaussian noise to a stationary arbitrary-sized texture image. This mechanism allows us to generate arbitrarily large textures while ensuring stationarity in output. We also find that InfinityGAN fails in conditional generation in most cases, where the one-hot class input does not precisely control the category of generated decay texture, while our method shows outstanding conditional generation ability.

2) DISENTANGLING STYLE AND RESOLUTION

Our decay effects generator uses the style code $\mathbf{Z}_{\mathbf{style}}$ to control the overall style and the $\mathbf{Z}_{\mathbf{res}}$ to define the resolution of the output. For investigating the disentanglement of those two inputs, we show texture results by switching different $\mathbf{Z}_{\mathbf{res}}$ while maintaining the same $\mathbf{Z}_{\mathbf{style}}$. The texture results are shown in FIGURE 8. From FIGURE 8, we find that the $\mathbf{Z}_{\mathbf{style}}$

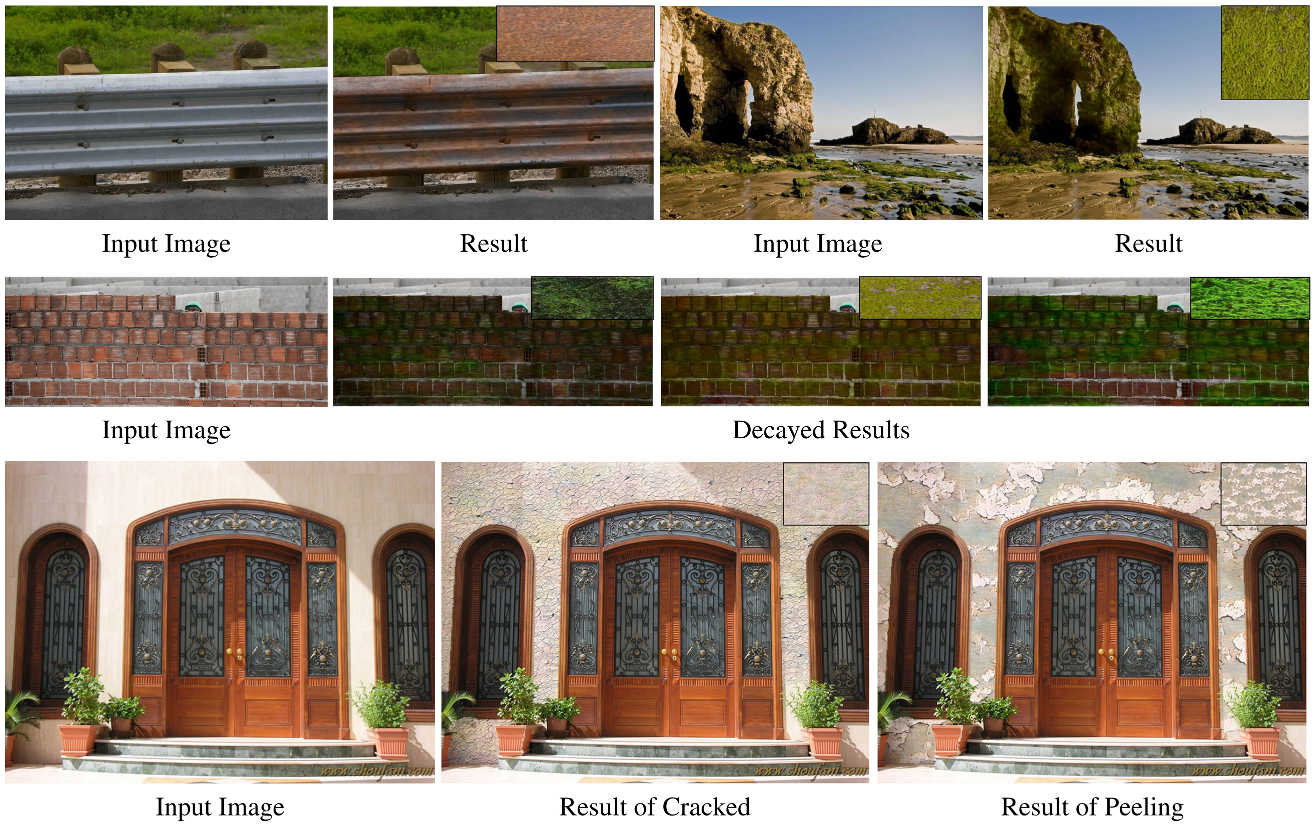


FIGURE 10. Decay results with our method. The first row shows varying decay results of different target objects. The second row shows diverse results of different categories of decay effects on the same object. The last row shows different results of the same category of decay effect for an object. Textures used for rendering are put in the upper-right corner or upper-left corner.

and the $\mathbf{Z_{res}}$ are properly disentangled, where $\mathbf{Z_{style}}$ controls the overall style and $\mathbf{Z_{res}}$ defines the resolution of the output.

3) DECAY EFFECTS SIMULATION

We evaluate our method using images from a broad range of scenes and compare it with the existing approaches. Since existing approaches cannot produce decay results exclusively, we incorporate these methods into our framework, where texture generation is replaced with existing methods. Comparisons between the results of our method and the existing method can be seen in FIGURE 9. We find that our framework produces more realistic decay results than existing methods. We argue that unrealistic results of the existing work are produced due to the poor conditional generation ability and non-stationary output. In addition to comparisons with existing methods, we also show diverse patterns of decay results produced by our method in FIGURE 10. We also show a decay result on complicated scenes with varying effects on different objects in FIGURE 11. The decay effects are initialized and conditioned with the user inputs. Our framework can also produce realistic and seamless results for partially decayed cases, as illustrated in FIGURE 12. This capability is ensured through the integration of Perlin noise initialization (Section III-C2) and our decay diffusion strategy (Section III-C4).

C. QUANTITATIVE EVALUATION

1) TEXTURE GENERATION

The results with the FID are shown in TABLE 3. The results of the InfinityGAN and our method at 2048 × 2048 pixels are generated in a patch-by-patch manner, while others are generated at one single forward pass. From TABLE 3, we observe that PSGAN and CT-PSGAN are far from generating realistic decay textures. Although InfinityGAN and our method both generate realistic textures, our method outperforms InfinityGAN in most cases.

The results with the stationarity metric are shown in TABLE 4. We report the average value of 4,000 textures generated by each method at different resolutions. We omit the results of the PSGAN and the CT-PSGAN because of the poor quality of generated textures. From TABLE 4, we can observe that textures generated by our generator show more stationarity than textures generated by InfinityGAN. Moreover, our decay effects generator maintains a consistent performance at varying resolutions of textures.

2) DECAY EFFECTS SIMULATION

We further conduct a perceptual user study to quantitatively verify the effectiveness of our framework. Similar to qualitative comparison, we also incorporate other generative models into our framework. We use 57 images for evaluation

TABLE 3. Comparisons between our method against other texture generation models at different sizes, in terms of FID. The results of the InfinityGAN [15] and our method at 2048 × 2048 pixels are generated in a patch-by-patch manner, while others are generated at one single forward pass.

Method	128×128	256×256	512×512	1024×1024	2048×2048
PSGAN [43]	297.61	344.50	323.47	427.26	366.71
CT-PSGAN [43], [51]	159.84	184.54	244.44	263.93	342.02
InfinityGAN [15]	**75.82**	48.18	72.42	132.37	183.76
Ours	78.09	**47.81**	**55.78**	**100.12**	**157.92**

TABLE 4. Comparisons between our method against InfinityGAN at different sizes, in terms of metric of the stationarity. The results of the PSGAN and CT-PSGAN are omitted since they failed at generating convincing textures. Our decay effects generator surpasses InfinityGAN and maintains a consistent performance at varying resolutions of texture images.

Method	128×128	256×256	512×512	1024×1024	2048×2048
InfinityGAN [15]	0.3033	0.7013	1.7007	6.2060	5.8820
Ours	**0.2931**	**0.3555**	**0.3459**	**0.4334**	**0.4564**

TABLE 5. Ablation results of our decay effects generator on decay texture generation in terms of FID. Conditional StyleGAN2 can only generate textures at a fixed size. Ours w/o consistent transformation (CT) cannot generate large images because it consumes a huge amount of memory. Our full method generates stationary decay textures with seamless incremental generation.

Method	128×128	256×256	512×512	1024×1024	2048×2048
Conditional StyleGAN2 [16]	66.94	-	-	-	-
Ours w/o CT	**66.33**	**42.73**	69.29	116.50	OOM
Ours	78.09	47.81	**55.78**	**100.12**	**157.92**

TABLE 6. Perceptual user study result. The numbers indicate the percentage of the images that are deemed to be real over the other method.

Method	Naturalness
InfinityGAN [15]	31.02%
Ours	**68.98%**

and generate decay effects with two different approaches, including InfinityGAN and our method. We manually specify a class label for each image before the simulation. As for the degree decision used in the user study, we simulate a most-decayed case, where the density is 80% and the continuity is high. Afterward, we obtain 20 random decay results for each image by each approach. In total, we have 2,280 decay results that are generated from 57 natural images by two different approaches. We invite 10 subjects to participate in this study, given the input image as a reference, the subjects are asked to pick the more realistic one among the results produced by the two approaches. As shown in TABLE 6, our framework is considered to be better in 68.98% of the cases.

In our quantitative evaluation, we compare our approach with three different methods, as detailed in TABLE 3. Among these, only InfinityGAN is able to generate relatively realistic textures. Consequently, our subsequent evaluations (TABLE 4 and TABLE 6) focused primarily on comparisons with InfinityGAN. This decision ensured accurate and unbiased quantitative results. Our evaluation choices are designed to provide a clear, rigorous, and fair assessment of our method's capabilities.

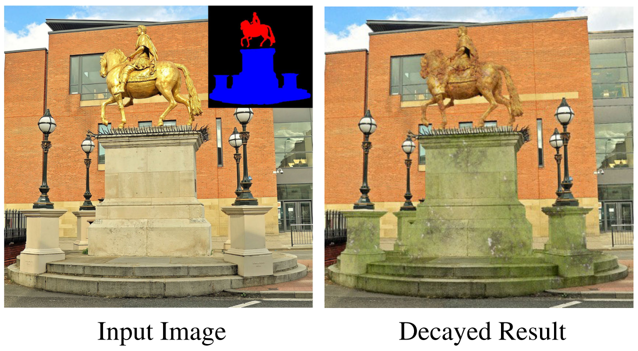


FIGURE 11. Decayed image with multiple decay effects. Our decay framework is able to process complicated images with multiple effects on different objects. The decay effects are initialized and conditioned with user inputs. The user inputs are visualized as colored shapes.

D. ABLATION STUDY

We perform an ablation study to demonstrate the effectiveness of the proposed network in terms of FID on texture generation. Our model is based on a conditional StyleGAN2 that achieves conditional texture generation by concatenating a one-hot class label with the style input. To generate stationary decay textures, we replace the learned constant input of the StyleGAN2 with a white Gaussian noise input (ours without consistent transformation). Our full method, formed by consistent transformation, achieves seamless results in an incremental generation fashion, that is, separate patches of the output are generated individually and composited seamlessly. The ablation results are shown in TABLE 5. Note that ours without consistent transformation

Algorithm 1 Pseudocode of the Initialization of Degree Map in a Python-Like Style

```
# continuity = 0.06: parameter controls the continuity. The lower it is, the more continuous the output will be.
# maks_area = 0.25: parameter defines the density of the mask region.
# image_size: the size of the input image
# max_size = 10000

# low_res_pattern sampled from a uniform distribution
low_res_pattern = np.random.uniform(0, 1, (max_size * continuity, max_size * continuity) * 255)
# resize the low_res_pattern to the max_size to construct a high-resolution pattern
pattern = cv2.resize(low_res_pattern, dsize=(max_size, max_size), interpolation=cv2.INTER_CUBIC)
# construct binary mask with respect to mask_area
pattern = pattern / 255
pattern = np.less(pattern, mask_area)
# mask area refinement
while True:
x, y = np.random.randint(0, pattern.size()[0] - image_size, 2)
mask = pattern[x: x+image_size, y: y+image_size]
pattern_mask_area = mask.float().mean().item()
# If mask is whthin +/- 25\% of desired mask area, break
if mask_area / 1.25 < pattern_mask_area < mask_area * 1.25
```

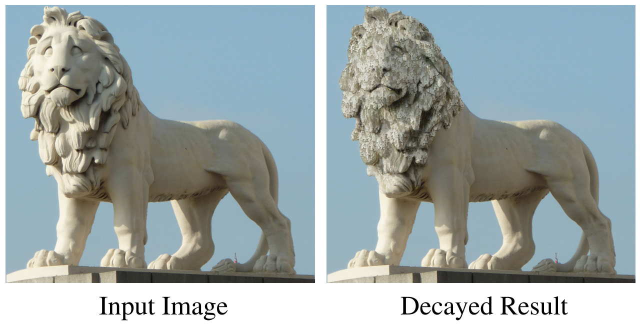


FIGURE 12. Partially decayed results. Our decay framework can produce realistic and harmonious outputs in partial decay cases. This seamless appearance is ensured by the Perlin noise with our decay diffusion strategy .

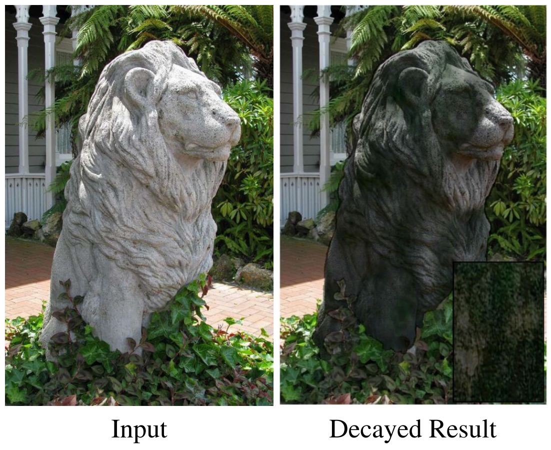


FIGURE 13. Failure case produced by our framework. Our framework may generate an obviously unrealistic decay texture for the input image.

(Ours w/o CT) raises an out-of-memory (OOM) error on common GPUs when generating the size of 2,048 $\times$ 2,048 as it does not support incremental generation.

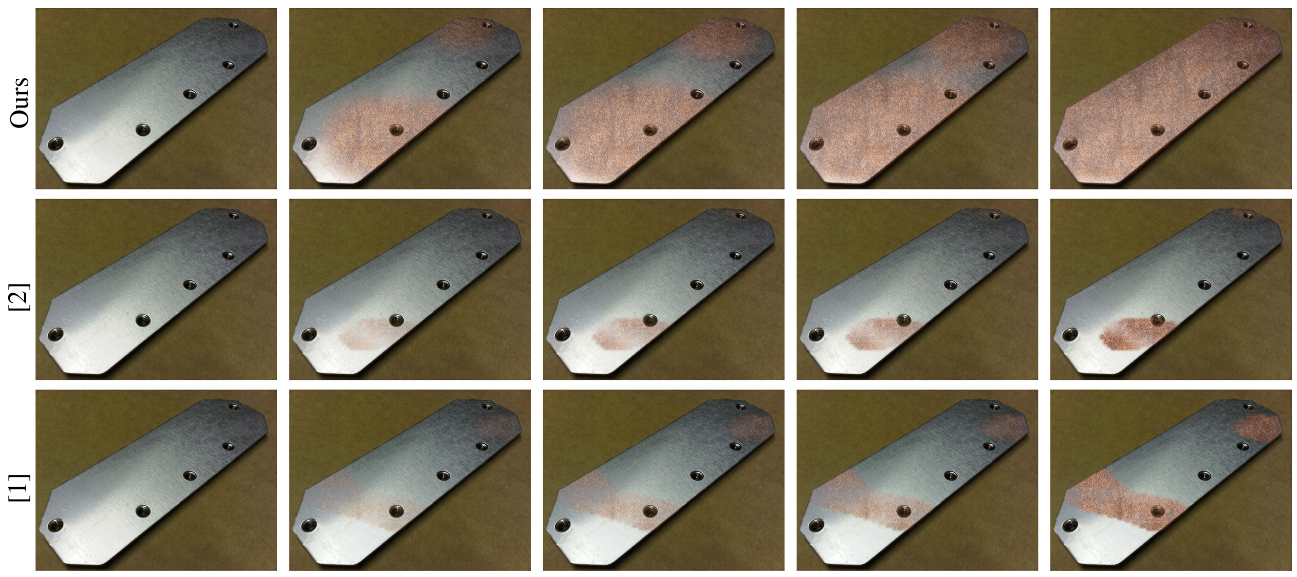
E. LIMITATIONS

Although our framework is able to generate plausible results in most cases, it has several limitations. Unfavorable results, such as FIGURE 13, are produced by using an obviously unrealistic decay texture with respect to the input image. To deal with such a case, we can generate another decay texture by exploring a different style vector as our framework runs fast. Also, our framework is not able to propagate decay effects for a decayed image. In this case, we consider propagating decay effects by inversing decay regions through our decay effects generator via GAN inversion techniques [58] or simply using the existing method such as patch-based weathering propagation [4].

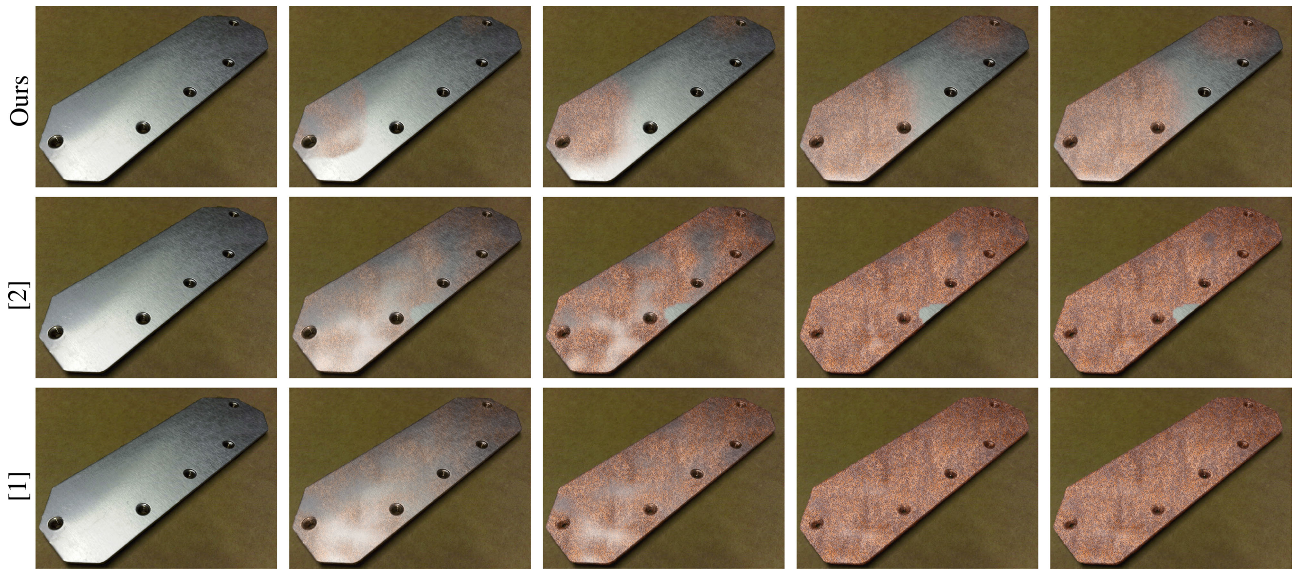
V. CONCLUSION AND FUTURE WORK

We proposed a novel framework for simulating time-varying decay effects in natural images. The proposed framework can generate diverse decay effects on non-decayed images without referencing other decayed images, which is impossible with existing image-based decay methods. One of the key ideas for the success of our approach is a stable decay effect generation mechanism that is able to generate stationary and arbitrary-sized decay textures. Furthermore, we created a decay texture dataset including major categories of real-world decay to train the decay generator. We believe our work makes significant progress in the task of image-based decay by enlarging the application range. We also conducted in-depth evaluations to demonstrate the effectiveness of our proposed framework.

In future work, we would like to extend our framework to propagate decay effects for a decayed image through the GAN inversion techniques [58]. In detail, decayed regions from an input image are first inversed to latent codes of the decay effects generator. And then, we modify the size of flexible-sized Gaussian noise to generate a decay texture that matches the resolution of the input image. Also, we would like to automatically detect the object regions for an input image. So far, the object region is obtained by a user-provided mask, which requires expertise and is time-consuming. Furthermore, an interesting avenue for future research would be to integrate the effects of structural decay, such as rock weathering. This would require a combined approach addressing both texture variations and structural



(a) Less-decayed initialization.



(b) Most-decayed initialization.

FIGURE 14. Visual comparisons between our proposed decay effects diffusion strategy and other propagation strategies under less and most decayed initialization.

deterioration, providing a more comprehensive simulation of realistic natural decay.

APPENDIX A
RANDOM PATTERN GENERATION

In this section, we provide pseudocode 1 for the random pattern generation described in Section III-C2.

APPENDIX B
DECAY EFFECTS DIFFUSION

To validate our decay diffusion approach detailed in Section III-C4, we compare it with decay propagation techniques from prior image-based decay studies [1], [2]. Traditional methods derive an initial degree map from decayed images. However, it is impossible to derive an initial degree map from non-decayed images. To address this, we utilize our degree map initialization method (Section III-C4) to provide an initial map for all decay propagation techniques. We then compare our method against time-dependent function [1] and smooth expansion [2], using both less and most decayed maps for initialization. Results in FIGURE 14(a) and FIGURE 14(b) reveal that while traditional methods only effectively propagate from highly decayed cases, our approach succeeds in both varying decay situations and in extending from decayed to non-decayed areas.

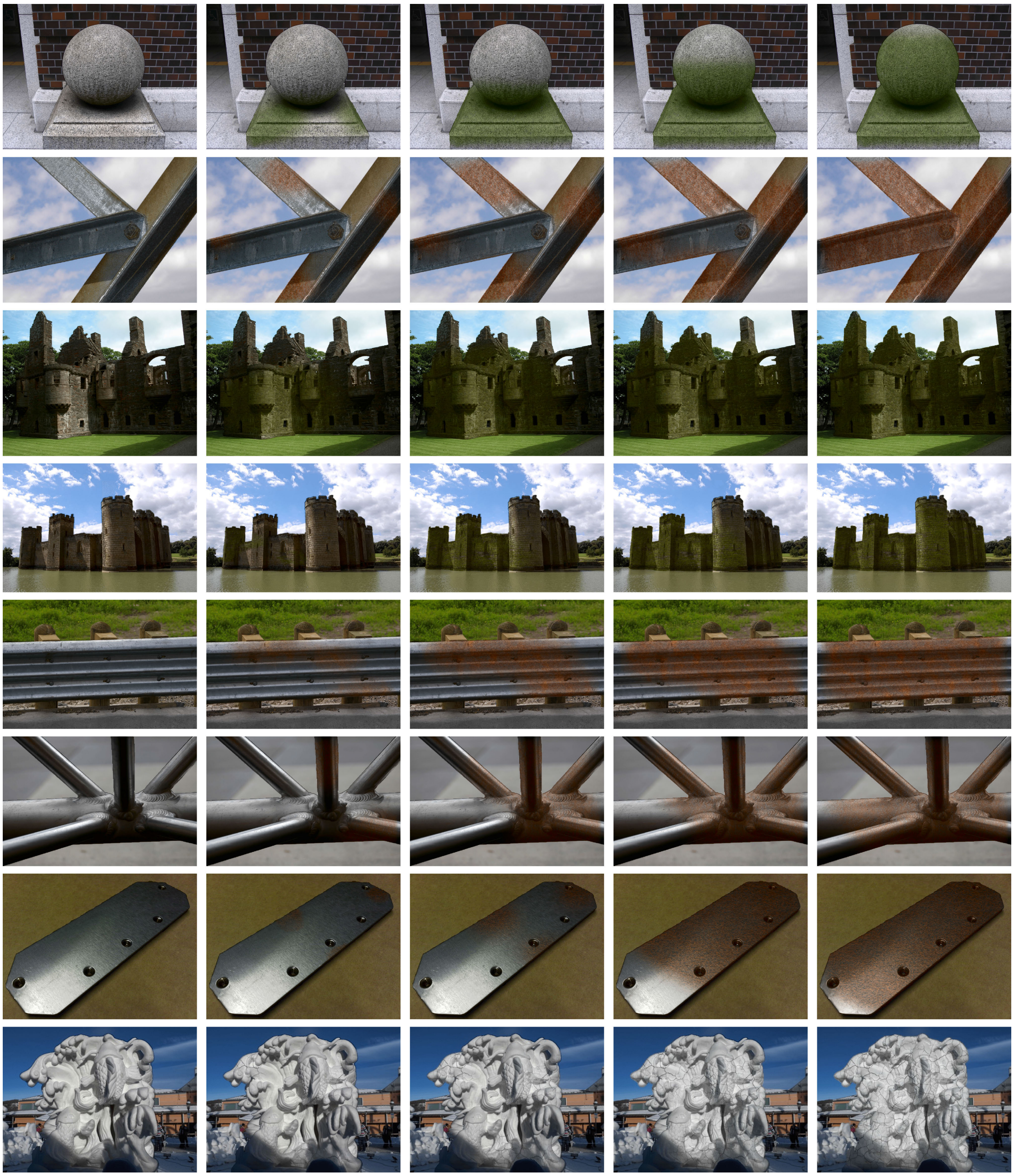


FIGURE 15. Time-varying decay results produced by our proposed framework. The first image in each row is the input image.

APPENDIX C
DECAY EFFECTS SIMULATION

In this section, we present additional time-varying decay results produced by our framework in FIGURE 15.

REFERENCES

[1] S. Xuey, J. Wang, X. Tong, Q. Dai, and B. Guo, ''Image-based material weathering,'' *Comput. Graph. Forum*, vol. 27, no. 2, pp. 617–626, Apr. 2008.

[2] D. Bandeira and M. Walter, "Synthesis and transfer of time-variant material appearance on images," in *Proc. 22nd Brazilian Symp. Comput. Graph. Image Process.*, Oct. 2009, pp. 32–39.

[3] Y. Endo, Y. Kanamori, J. Mitani, and Y. Fukui, "Image editing for weathering effects with geometric details," in *Proc. Comput. Graph. Int. (CGI)*, 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:5616958

[4] S. Iizuka, Y. Endo, Y. Kanamori, and J. Mitani, "Single image weathering via exemplar propagation," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 501–509, May 2016.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5967–5976.

[6] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2242–2251.

[7] Z. Wang, J. Chen, and S. C. H. Hoi, "Deep learning for image super-resolution: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3365–3387, Oct. 2021.

[8] W. Zhang, P. Zhuang, H.-H. Sun, G. Li, S. Kwong, and C. Li, "Underwater image enhancement via minimal color loss and locally adaptive contrast enhancement," *IEEE Trans. Image Process.*, vol. 31, pp. 3997–4010, 2022.

[9] W. Zhang, L. Zhou, P. Zhuang, G. Li, X. Pan, W. Zhao, and C. Li, "Underwater image enhancement via weighted wavelet visual perception fusion," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Jul. 27, 2023, doi: 10.1109/TCSVT.2023.3299314.

[10] W. Li, Z. Lin, K. Zhou, L. Qi, Y. Wang, and J. Jia, "MAT: Mask-aware transformer for large hole image inpainting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10748–10758.

[11] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10674–10685.

[12] C. Saharia, W. Chan, H. Chang, C. A. Lee, J. Ho, T. Salimans, D. J. Fleet, and M. Norouzi, "Palette: Image-to-image diffusion models," in *Proc. ACM SIGGRAPH Conf.*, 2022, doi: 10.1145/3528233.3530757.

[13] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020, *arXiv:2006.11239*.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Conf. Neural Inf. Process. Syst.*, 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[15] C. H. Lin, H.-Y. Lee, Y.-C. Cheng, S. Tulyakov, and M.-H. Yang, "InfinityGAN: Towards infinite-pixel image synthesis," in *Proc. Int. Conf. Learn. Represent.*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:238419701

[16] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8110–8119.

[17] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6629–6640.

[18] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," 2015, *arXiv:1508.06576*.

[19] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Proc. Conf. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 262–270.

[20] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Computer Vision—ECCV 2016* (Lecture Notes in Computer Science), vol. 9906, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, doi: 10.1007/978-3-319-46475-6_43.

[21] R. Bellini, Y. Kleiman, and D. Cohen-Or, "Time-varying weathering in texture space," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–11, Jul. 2016.

[22] L.-Y. Chen, I.-C. Shen, and B.-Y. Chen, "Guided image weathering using image-to-image translation," in *Proc. SIGGRAPH Asia Tech. Commun.*, Dec. 2021, doi: 10.1145/3478512.3488603.

[23] J. Dorsey and P. Hanrahany, "Modeling and rendering of metallic patinas," in *Proc. ACM SIGGRAPH Courses*, 2006, p. 2-es.

[24] J. Dorsey, H. K. Pedersen, and P. Hanrahan, "Flow and changes in appearance," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn.*, 1996, pp. 411–420.

[25] J. Dorsey, A. Edelman, H. W. Jensen, J. Legakis, and H. Pedersen, "Modeling and rendering of weathered stone," in *Proc. 26th Annu. Conf. Comput. Graph. Interact. Techn.*, 1999, pp. 225–234.

[26] E. Paquette, P. Poulin, and G. Drettakis, "The simulation of paint cracking and peeling," in *Proc. Graph. Interface Conf.*, 2002, pp. 59–68.

[27] C. Bosch, X. Pueyo, S. Mérillou, and D. Ghazanfarpour, "A physically-based model for rendering realistic scratches," *Comput. Graph. Forum*, vol. 23, no. 3, pp. 361–370, Sep. 2004.

[28] B. Desbenoit, E. Galin, and S. Akkouche, "Simulating and modeling lichen growth," *Comput. Graph. Forum*, vol. 23, no. 3, pp. 341–350, Sep. 2004.

[29] A. Ishitobi, M. Nakayama, and I. Fujishiro, "Visual simulation of weathering coated metallic objects," *Vis. Comput.*, vol. 36, nos. 10–12, pp. 2383–2393, Oct. 2020.

[30] A. Ishitobi, M. Nakayama, and I. Fujishiro, "Visual simulation of crack and bend generation in deteriorated films coated on metal objects: Combination of static fracture and position-based deformation," *Vis. Comput.*, vol. 39, no. 8, pp. 3403–3415, Aug. 2023.

[31] S. Merillou, J.-M. Dischler, and D. Ghazanfarpour, "Corrosion: Simulating and rendering," in *Proc. Graph. Interface Conf.*, Jun. 2001, pp. 167–174.

[32] J. Gu, C.-I. Tu, R. Ramamoorthi, P. Belhumeur, W. Matusik, and S. Nayar, "Time-varying surface appearance: Acquisition, modeling and rendering," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 762–771, Jul. 2006.

[33] J. Lu, A. S. Georghiades, A. Glaser, H. Wu, L.-Y. Wei, B. Guo, J. Dorsey, and H. Rushmeier, "Context-aware textures," *ACM Trans. Graph.*, vol. 26, no. 1, p. 3-es, Jan. 2007.

[34] B. Sun, K. Sunkavalli, R. Ramamoorthi, P. N. Belhumeur, and S. K. Nayar, "Time-varying BRDFs," *IEEE Trans. Vis. Comput. Graphics*, vol. 13, no. 3, pp. 595–609, May 2007.

[35] J. Wang, X. Tong, S. Lin, M. Pan, C. Wang, H. Bao, B. Guo, and H.-Y. Shum, "Appearance manifolds for modeling time-variant appearance of materials," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 754–761, Jul. 2006.

[36] S. Du and Y. Song, "Multi-exemplar-guided image weathering via texture synthesis," *Vis. Comput.*, vol. 39, no. 8, pp. 3691–3699, Aug. 2023.

[37] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the art in example-based texture synthesis," in *Proc. Eurographics, State Art Rep., EG-STAR*, Mar. 2009, pp. 93–117.

[38] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Techn.*, Aug. 2001, pp. 341–346.

[39] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 795–802, Jul. 2005.

[40] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3606–3613.

[41] D. Dai, H. Riemenschneider, and L. V. Gool, "The synthesizability of texture examples," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3027–3034.

[42] N. Jetchev, U. M. Bergmann, and R. Vollgraf, "Texture synthesis with spatial generative adversarial networks," 2016, *arXiv:1611.08207*.

[43] U. Bergmann, N. Jetchev, and R. Vollgraf, "Learning texture manifolds with the periodic spatial GAN," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 469–477.

[44] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proc. Int. Conf. Learn. Represent.*, 2018. [Online]. Available: https://openreview.net/forum?id=Hk99zCeAb

[45] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4217–4228.

[46] R. Xu, X. Wang, K. Chen, B. Zhou, and C. C. Loy, "Positional encoding as spatial inductive bias in GANs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13564–13573.

[47] E. Ntavelis, M. Shahbazi, I. Kastanis, R. Timofte, M. Danelljan, and L. Van Gool, "Arbitrary-scale image synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11523–11532.

[48] C. H. Lin, C.-C. Chang, Y.-S. Chen, D.-C. Juan, W. Wei, and H.-T. Chen, "COCO-GAN: Generation by parts via conditional coordinating," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4511–4520.

[49] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Diversified texture synthesis with feed-forward networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 266–274.

[50] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. Int. Conf. Learn. Represent.*, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:11758569
[51] C. Lu, R. E. Turner, Y. Li, and N. Kushman, "Interpreting spatially infinite generative models," in *Proc. 37th Int. Conf. Mach. Learn. Workshop Hum. Interpretability (WHI)*, 2020. [Online]. Available: https://arxiv.org/abs/2007.12411
[52] S. Bell, K. Bala, and N. Snavely, "Intrinsic images in the wild," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–12, Jul. 2014.
[53] Q. Mao, H.-Y. Lee, H.-Y. Tseng, S. Ma, and M.-H. Yang, "Mode seeking generative adversarial networks for diverse image synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 1429–1437.
[54] G. Georgiadis, A. Chiuso, and S. Soatto, "Texture compression," in *Proc. Data Compress. Conf.*, Mar. 2013, pp. 221–230.
[55] P. Kraehenbuehl and V. Koltun, "Parameter learning and convergent inference for dense random fields," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 513–521.
[56] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Comput. Graph. Appl.*, vol. 21, no. 4, pp. 34–41, 2001.
[57] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015. [Online]. Available: http://arxiv.org/abs/1409.1556
[58] W. Xia, Y. Zhang, Y. Yang, J.-H. Xue, B. Zhou, and M.-H. Yang, "GAN inversion: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3121–3138, Mar. 2023.

GUOQING HAO received the M.S. degree from the Department of Computer Science, University of Tsukuba, Japan, in 2021, where he is currently pursuing the Ph.D. degree. He is a Research Assistant with the National Institute of Advanced Industrial Science and Technology, Japan. His research interests include image processing, computer vision, and computer graphics.

SATOSHI IIZUKA received the Ph.D. degree in engineering from the University of Tsukuba. He is currently an Associate Professor with the Faculty of Engineering, Information and Systems, University of Tsukuba. His research interests include computer graphics and vision, including image processing and editing based on machine learning.

KENSHO HARA received the B.E. degree in information engineering, the M.S. degree in information science, and the Ph.D. degree in information science from Nagoya University, in 2012, 2014, and 2017, respectively. He is currently a Research Scientist with the National Institute of Advanced Industrial Science and Technology. His research interest includes video action recognition.

HIROKATSU KATAOKA received the Ph.D. degree in engineering from Keio University, in 2014. He is currently a Senior Researcher with the National Institute of Advanced Industrial Science and Technology (AIST). His research interests include computer vision and pattern recognition, especially in large-scale dataset for image and video recognition. He has received the ACCV 2020 Best Paper Honorable Mention Award, the AIST 2019 Best Paper Award, and the ECCV 2016 Workshop Brave New Idea.

KAZUHIRO FUKUI (Member, IEEE) received the Ph.D. degree from the Tokyo Institute of Technology, in 2003. He joined the Toshiba Corporate Research and Development Center. He was a Senior Research Scientist with the Multimedia Laboratory. He is currently a Professor with the Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba. His research interests include the theory of machine learning, computer vision, pattern recognition, and their applications. He is a member of the SIAM.