EUROGRAPHICS 2016 / J. Jorge and M. Lin
(Guest Editors)

Volume 35 (2016), Number 2

Single Image Weathering via Exemplar Propagation

Satoshi Iizuka[1] Yuki Endo[2] Yoshihiro Kanamori[2] Jun Mitani[2]

[1]Waseda University
[2]University of Tsukuba

Abstract

This paper presents an efficient approach for generating weathering effects with detailed appearance variations in a single image. Previous approaches merely change chroma or reflectance of weathered objects, which is not sufficient for materials with detailed shading and texture variations, such as growing moss and peeling plaster. Our method propagates such detailed features via seamless patch-based synthesis driven by weathering degree distribution. Unlike previous methods, the weathering degrees are calculated efficiently using Radial Basis Functions even for materials with wide color variations. We use graph cut-based optimization to identify the most weathered region as a "weathering exemplar", from which we sample weathering patches. We demonstrate our method enables us to generate various types of detailed weathering effects interactively.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—

1. Introduction

In the real world, most objects decay over time and change their appearance according to their material properties and surrounding environment. Such weathering effects are important factors to enhance realism of scenes in many applications in computer graphics. Previous weathering methods [WTL*06, XWT*08, EKMF11] assume that weathered materials such as fallen leaves and stains merely change their colors or reflectance in dichromatic transition. However, in the real world, there are a variety of weathered materials that exhibit transitions of not only their reflectance but also detailed shading or textures such as peeling plaster and growing moss (see Figure 1), to name but a few. Also, time-varying materials can have wider color distributions than dichromatic ones, just like the example of stone with lichen in Figure 4.

In this paper, we propose a single-image weathering technique that overcomes the above problems. Instead of assuming the weathering process as time-varying reflectance transition, we model weathering as a propagation process of detailed weathering features from highly-weathered regions. We estimate weathering degrees using *Radial Basis Functions* (RBFs) so that we can handle wide color distributions of weathered objects, unlike existing methods. We then extract a rectangular region with high weathering degrees as a "weathering exemplar" using graph cut. We sample weathering patches from the exemplar and spread them seamlessly via global optimization.

In a simulation step, we update the weathering degree map to control weathering effects. We assume that many types of weathering spread gradually around already-weathered regions, and thus we employ a simple filtering approach by default to smoothly dilate weathered regions over time. Also, our method can be easily incorporated in other image-based weathering simulations and image editing tools, e.g., flow stain [EKMF10], lichen growth [DGA04], and brush painting.

(a) Plaster peeling (b) Mossy growing

Figure 1: *Weathering effects with complicated feature variations.*

We demonstrate results of weathering effects produced by our methods including lichen growth, plaster peeling, metal rusting, and flower bloom. We also show a result of weathering transfer that replaces weathering effects from a reference image to another.

2. Related work

There have been various research efforts aimed at generating weathering effects using physically-based simulation. These techniques create realistic weathering effects for specific scenes, such as flow stain [DPH96], stone erosion [DEJ*99], metallic patinas [DH96], corrosion of metal [DH96], paint peeling [PPD02], dust accumulation [HW95], and lichen growth [DGA04]. However, they require accurate object geometry with specialized physical parame-

(a) De-weathering (b) Input image and user inputs (c) Weathering (d) Further weathering

Figure 2: *Image weathering and de-weathering using our method. Left to right: (a) a result of de-weathering; (b) the original image and user inputs (the white/black scribbles in the red circles indicate weathered/non-weathered pixels); and (c)(d) results at different stages of weathering. User inputs in our other results are also marked with red circles.*

ters that are hard to estimate from a single image. Other techniques [GTR*06, LGG*07, SSR*07] captured time-varying reflectance of materials to produce temporal variations of appearance. Although these methods can create several visually-convincing weathering effects, they cannot be applied to materials in a single image due to their requirement of capturing full-time appearance transition of target materials.

In contrast to the above approaches, a variety of methods have been proposed for image-based material weathering. Xue et al. [XDR11] produced weathering effects on stones by modifying the appearance of their geometry. Endo et al. [EKMF10] and Bosch et al. [BLR*11] proposed systems for generating flow stain on objects based on particle simulation. These methods achieve convincing results by limiting target materials and weathering effects.

For more general modeling of weathering effects, Wang et al. [WTL*06] presented a visual simulation method that acquires time-varying appearance from BRDFs at various points on a single material. They construct an *appearance manifold*, which approximates the appearance subspace caused by weathering to compute a distribution of weathering degrees and simulate the temporal variation of material surfaces. Whereas this algorithm mainly targets 3D models, Xue et al. [XWT*08] targeted to generate weathering effects in 2D images by incorporating a decomposition scheme for modeling reflectance variations. Bandeira and Walter [BW09] proposed an *appearance map*, which simplifies the appearance manifold into two-dimensional feature space and dramatically reduces computational costs. Endo et al. [EKMF11] refined results of the appearance map by adding high-frequency components for representing shading variations. These methods can generate visually-natural weathering effects in some specific cases where weathered objects gradually undergo dichromatic color transition. Consequently, these algorithms fail to change reflectance with wide color variations because chroma distribution converge into a single color of the most weathered point. Additionally, their methods cannot produce complicated appearance variations (e.g., texture patterns) because they change only reflectance or add high-frequency components for representing weathering effects. In our method, we model reflectance with such wide color variation using *Radial Basis Functions*, with which we can plausibly compute the distribution of weathering degrees. Then, we propagate the appearance features by sampling patches from salient weathering exemplars and pasting them on target positions based on both the weathering degrees and appearance coherency. This algorithm can generate the detail-aware weathering effects that cannot be produced by existing approaches.

Texture synthesis and completion. The steps of sampling and pasting patches in our approach are related to texture synthesis and image completion. A comprehensive review of these methods is out of our focus, and we only refer to several methods that have important relationships with our method. Efros and Freeman [EF01] proposed a texture synthesis technique based on stitching of texture patches. This algorithm is improved in several directions, such as graph cut-based stitching [KSE*03] and energy optimization [KEBK05, HZW*06]. For image completion, Wexler et al. [WSI07] and Simakove et al. [SCSI08] proposed a global optimization-based technique that can obtain more consistent fills. Barnes et al. accelerated these methods by a fast randomized patch search algorithm called PatchMatch [BSFG09, BSGF10]. Darabi et al. [DSB*12] demonstrated more generalized patch-based optimization by integrating image gradients into the patch representation. Although these methods achieve convincing fills with global consistency, they may fail when holes are large against sampling regions of patches. Our synthesis of weathering effects can be broadly regarded as an image completion problem, where non-weathered regions are holes and weathered regions are exemplars. Due to the drawbacks mentioned above, existing techniques for texture synthesis or image completion will fail if we apply them directly because weathered regions are often sparse and non-weathered regions are large against the weathered regions. Even worse, if we extract weathered regions with a threshold, they have many small holes with complicated boundaries, and thus cannot be directly used for synthesis. In this paper, we first generate a hole-free weathering exemplar from the most weathered pixels. We can successfully compute textures to cover non-weathered regions using the exemplar via patch-based optimization that enables seamless transition from highly-weathered regions while maintaining neighboring-patch coherence.

3. Image weathering with complicated feature variation

An overview of our method is illustrated in Figure 3 and Algorithms 1 and 2. Given an input image, the user extracts target objects using, e.g., GrabCut [RKB04] or a general image editing tool. For the object region, the user specifies the most and least weath-

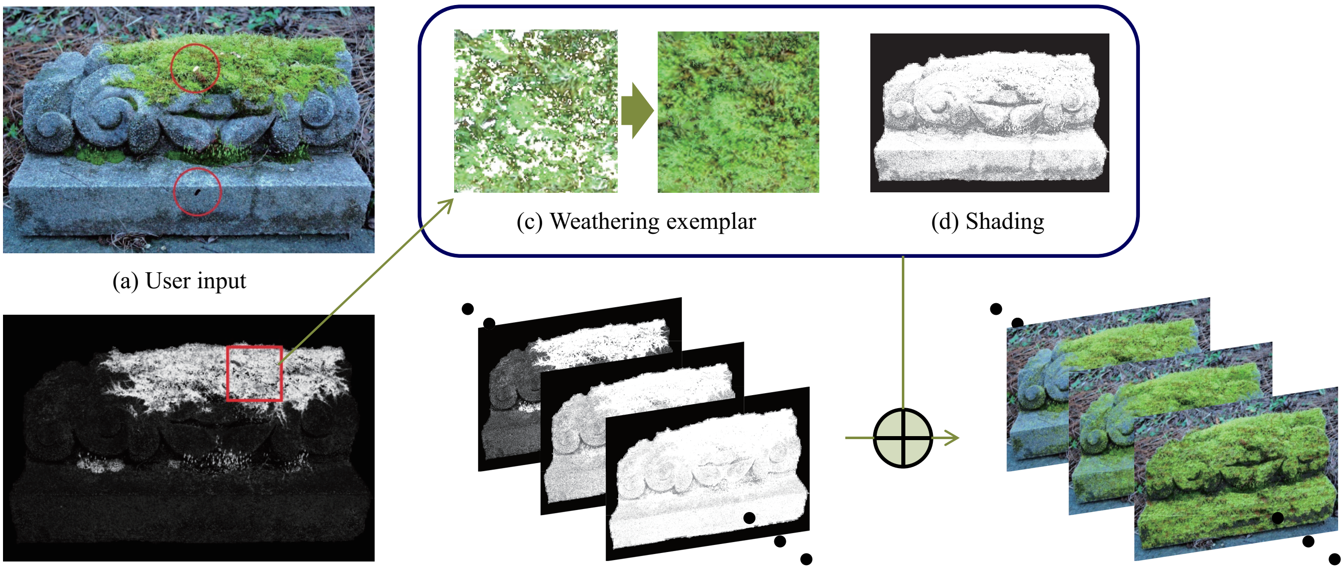


Figure 3: *Overview of our method. (a) Given an object region of the input image, the user specifies the most weathered pixels (white) and least weathered pixels (black) with scribbles. (b) The weathering degree map is then computed based on Radial Basis Function interpolation. (c) The highly-weathered region is extracted using graph cut and the weathering exemplar is generated by filling the holes. At the same time, (d) the shading map is computed. (e) When the weathering degree map is updated, (f) weathering patches sampled from the exemplar and the shading are composed according to the degree map.*

Algorithm 1 Computing weathering properties

Inputs: input image I and the most/least weathered pixels Ω (Option: mask that defines object regions to be weathered)
Outputs: weathering degree map D, global shading S, and weathered texture Z

$\alpha \leftarrow \mathrm{argmin}_\alpha E(\alpha)$; // Eq. 1
for each pixel $p \in I$ **do**
  $\mathbf{f}_p \leftarrow \textsc{FeatureVector}(L_p, a_p, b_p, x_p, y_p, \sigma_a, \sigma_s)$;
  $D_p \leftarrow \textsc{InterpolateUsingRBFs}(\mathbf{f}_p, \mathbf{f}_{i \in \Omega}, \alpha)$; // Eq. 2
end for
Labels $L \leftarrow \mathrm{argmin}_L E(L, D)$; // Eq. 3 (Sec. 3.3.1)
$R \leftarrow \textsc{FindRectangleWithSAT}(L)$; // Sec. 3.3.1
$T \leftarrow \textsc{MakeExemplarWithHoleFilling}(I, R, L)$;
$S \leftarrow \textsc{ComputeGlobalShading}(I, D, L)$; // Sec. 3.2
$Z \leftarrow \textsc{OptimizeTextureTiles}(I, T, D, L)$; // Sec. 3.3.2

Algorithm 2 Weathering synthesis

Inputs: input image I, weathering degree map D, global shading S, and weathered texture Z
Output: weathered image I'

$D \leftarrow \textsc{UpdateWeatheringDegree}(D)$; // simulation or manual editing (Sec. 3.5)
$D' \leftarrow \textsc{RectifyWeatheringDegree}(D)$; // Sec. 3.4
$I' \leftarrow \textsc{SynthesizeWeatheredTexture}(I, D', S, Z)$;

ered pixels with scribbles (Figure 3(a)). We use these pixels as samples to construct a smooth function that interpolates the weathering degree in a high-dimensional feature space. We solve the function interpolation problem efficiently using *Radial Basis Functions* (RBFs). Using the constructed nonlinear function, a weathering degree map is computed immediately (Figure 3(b)). Then, the most weathered region is identified from highly-weathered pixels using graph cut. The extracted region often has various holes, and thus the holes are completed using the existing image completion technique to obtain a *weathering exemplar*, which consists of salient features of the most weathered region (Figure 3(c)). At the same time, global shading of the object is extracted based on the weathering degrees. The shading map roughly represents the surface details (Figure 3(d)). When we update the weathering degree map (Figure 3(e)), the weathered object is generated by propagating weathering patches sampled from the weathering exemplar via global optimization that preserves appearance consistency, guided by the degree map. Finally, the resultant image is obtained by combining the weathering effects with the global shading (Figure 3(f)). We explain details of the processes in the following sections.

3.1. Weathering degrees based on Radial Basis Functions

The function interpolation with RBFs has been used widely in computer graphics, such as 3D surface reconstruction from point-cloud data [CBC*01], edit propagation [LJH10], and material deformation [BBO*10]. In this paper, we represent distribution of weathering degrees as a smooth function in a high-dimensional feature space. This is based on an observation that weathering degrees are smoothly varying in feature space of both visual appearances and spatial locations. This approach can plausibly compute the distribution of weathering degrees with wide color variation, unlike the previous techniques based on the appearance manifold [WTL*06, XWT*08] and the appearance map [BW09, EKMF11], as discussed in Section 2. Additionally, the computational cost is significantly low because it depends on the number of RBFs centered at the user-specified pixels, for which only a few dozens or hundreds of pixels are required in our method.

To compute a weathering degree map, we first determine the coefficient of each RBF. A weathering degree of each pixel is calculated as linear combination of RBFs. Let Ω denote a set of pixels within the user-specified scribbles. We assign an RBF $\phi(r) = e^{-r^2}$ to each pixel $i \in \Omega$, and calculate its coefficient by solving the following least-squares problem:

$$E(\alpha) = \sum_{i\in\Omega}(d_i - \sum_{j\in\Omega}\alpha_j\phi(\|\mathbf{f}_i - \mathbf{f}_j\|))^2, \qquad (1)$$

where $\mathbf{f}_i = (L_i/\sigma_a, a_i/\sigma_a, b_i/\sigma_a, x_i/\sigma_s, y_i/\sigma_s)$ is a feature vector, which consists of a visual feature and a spatial feature. L_i, a_i and b_i are luminance and chroma values in Lab color space while x_i and y_i are normalized pixel coordinates. σ_a and σ_s are parameters for controlling effects of appearance and spatial locality. We use $\sigma_a = 0.2$ and $\sigma_s = 50$ based on our observation that weathering effects are distributed more widely in spatial space compared to color space. α_j are unknown coefficients and d_i is a weathering degree in each scribble, where the least weathered pixels have $d_i = 0.01$ and the most weathered pixels $d_i = 1$. Similarly to Li et al. [LJH10], we solve the optimization problem using non-negative least-squares [LH74] to compute the binding coefficient α_i.

Using the computed coefficients α, we calculate a weathering degree $D_p \in [0,1]$ of each pixel p:

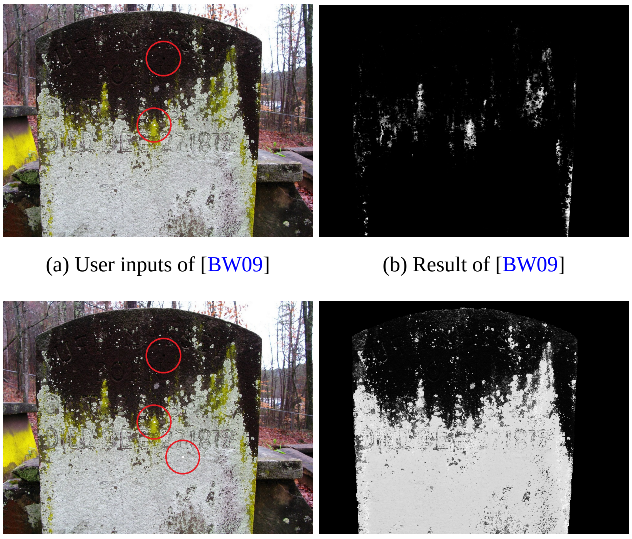
$$D_p = \sum_{i\in\Omega}\alpha_i\phi(\|\mathbf{f}_p - \mathbf{f}_i\|). \qquad (2)$$

Figure 4 illustrates a comparison of the weathering degree maps calculated using different methods. The appearance-based method [BW09] succeeds in identifying the brown stone as low degrees and the green lichen as high degrees. However, it fails to assign high degrees to the white lichen because the method bases only on two distinct pixels specified by the user as the least/most weathered. Alternatively, if the user selects the most weathered pixel in the white lichen, the green lichen will not be labeled as "weathered" but labeled as "non-weathered", which is strange. In contrast, our method can sample multiple distinct sets of pixels and construct a smooth function that interpolates weathering degrees in feature space, and therefore we can plausibly compute weathering degrees of the both green and white lichen.

3.2. Global shading

We seek to propagate appearance features of weathering effects using patch-based synthesis. However, directly pasting the weathering patches onto target positions sometimes erases the intrinsic shape of the underlying object. Because the intrinsic shape appears in the global shading, we can preserve the object shape by keeping the shading. Therefore, we compute global shading in advance and then compose it with the patches.

We compute the global shading based on weathering degrees similarly to [XWT*08, BW09]. In a weathered object with complicated appearance variation, it is difficult to decompose intrinsic shading of the object and detailed shading of weathering effects. However, for less-weathered regions, the intrinsic shading of the object seems to stem from differences of luminance components of pixels with the same weathering degrees. Based on this observation, we modify the shading computation in [XWT*08, BW09] by simply sampling pixels only from less-weathered regions. We utilize the result of binary segmentation described in Section 3.3.1 for identifying the less-weathered regions. We uniformly partition weathering degrees in the less-weathered region into 20 discrete intervals and calculate average luminance values μ_k for each interval k. Then, we divide luminance $I^{lum}_{p,k}$ of each pixel p in interval k by the corresponding average luminance μ_k to compute the global shading $S_{p,k}$ (Figure 3(d)), i.e., $S_{p,k} = I^{lum}_{p,k}/\mu_k$. The remaining highly-weathered regions keep original luminance because appearances of the regions remain unchanged during weathering simulation.



(a) User inputs of [BW09] (b) Result of [BW09]

(c) User inputs of ours (d) Result of ours

Figure 4: *Comparisons of weathering degree maps of [BW09] and our method. (a) Previous methods [WTL*06, XWT*08, BW09, EKMF11] only allow a single pixel for each of the least/most weathered points, e.g., the least weathered point in the stone and the most weathered point in the green lichen. Thus, (b) they fail to classify the white lichen as a weathered region. In contrast, (c) from multiple sets of pixels, (d) our method can continuously estimate weathering degrees for both green and white lichens.*

3.3. Patch-based weathering propagation

We generate weathering effects by propagating weathered features in a patch basis using weathering degrees, unlike the previous methods that change only chroma or reflectance. As briefly discussed in Section 2, directly using the existing texture synthesis techniques [BSFG09, SCSI08, WSI07, DSB*12] is difficult in our case because weathered regions are often sparse and have small holes.

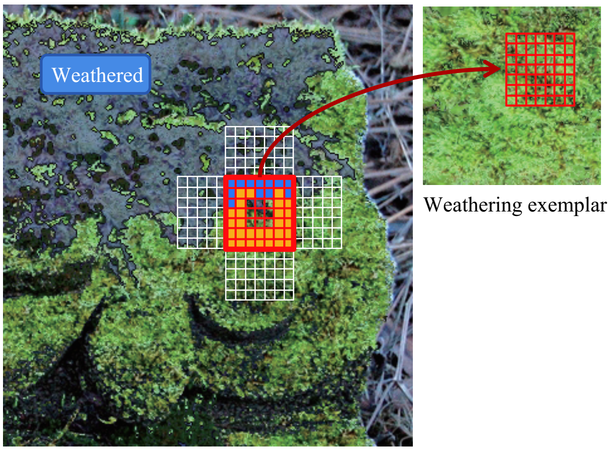


Figure 5: *Search of weathering patches. For the target patch (red grid), an optimal patch in weathering exemplar is computed based on differences of colors in overlapping pixels (orange blocks) with neighboring patches (white) and constant pixels (blue blocks).*

We generate such complicated weathering effects in the following processes. First, we identify the most salient weathered samples, called *weathering exemplar*, from highly-weathered regions using graph cut. Then, we find optimal patches of weathering via global energy optimization guided by weathering degrees. Finally, we synthesize the patches and global shading to generate weathering effects. We explain the details in the following sections.

3.3.1. Weathering exemplar

In our method, the weathering exemplar is defined as a rectangular region that includes the most salient weathered features. Given a weathering map, we first perform binary segmentation, similarly to [BJ01], to label weathered and non-weathered regions. Based on our knowledge that salient weathered pixels are often located densely in various natural images, we define a cost function $E(L)$ using the weathering degrees D:

$$E(L) = \sum_{p} -\log U_p(D_p, L_p) + \lambda \sum_{p,q \in \mathcal{N}} [L_p \neq L_q] V_p(D_p, D_q), \quad (3)$$

$$U_p(D_p, L_p) = \begin{cases} D_p & \text{if} \quad L_p \in \text{"weathered"} \\ 1 - D_p & \text{otherwise.} \end{cases}$$

where $L_p \in \{$"weathered", "non-weathered"$\}$ denotes binary labels, $\mathcal{N}$ is the set of pairs of neighboring pixels, and $[\cdot]$ denotes the indicator function taking 1 if the argument is true and 0 otherwise. V_p is a smoothness term and λ is a coefficient that determines the relative importance of the smoothness term. We use $V_p(D_p, D_q) = \exp(-\beta(D_p - D_q)^2 / dist(p,q))$, where $dist(p,q)$ is the Euclidean distance of neighboring pixels and $\beta = (2\langle (D_p - D_q)^2 \rangle)^{-1}$ is a constant value, where $\langle \cdot \rangle$ denotes the average, as described in [RKB04]. In our experiment, λ is set to 20 for all the results. We can efficiently solve the energy minimization problem using graph cut. Then, we extract a rectangular that includes the most pixels labeled as "weathered" (Figure 3(b), red). The size of the rectangular is 150×150 pixels in our system. This can be computed in real time using a summed area table (SAT) [Cro84]. The extracted region often has holes (Figure 3(c), left) that will cause noisy weathering effects. Furthermore, pixels inside the holes will be unchanged even when weathering degrees increase significantly. To avoid these problems, we fill them in using PatchMatch [BSFG09] (Figure 3(c), right).

(a) [BSFG09] (b) [DSB*12] (c) Ours

Figure 6: *Comparison of synthesis of weathering effects. (a)(b) While existing methods generate visually-inconsistent texture, (c) our method generates detailed weathered texture.*

3.3.2. Weathering patch search via global optimization

After generating the weathering exemplar, we spread weathering patches sampled from it to the whole region of the target object. In this process, we keep the pixels with label "weathered" unchanged because they are considered as weathered completely. To synthesize texture-preserving weathering effects, we compute weathering patches in an input image I, which are sampled from the weathering exemplar T in the following steps. First, we randomly sample patches from the weathering exemplar and tile them so that they have overlaps with neighboring patches, similarly to the image quilting techniques [EF01, KEBK05]. In our system, the width of the overlap on one side is $1/4$ of the tile size. Then, we iteratively update the patches based on a global distance measure $d(I,T)$. We show a schematic illustration in Figure 5. For each patch (red grid) in the input image, we search for optimal patches that minimize SSD (sum of squared differences) of colors in overlapping pixels (orange pixels) with neighboring patches (white grid) and pixels labeled "weathered" (blue pixels). $d(I,T)$ is defined as follows:

$$\begin{aligned} d(I,T) = \; & \textstyle\sum_{p \subset I} \min_{q \subset T} (\mathcal{K}_p G_{dat}(p,q) \\ & + (1 - \mathcal{K}_p) G_{neighbor}(p,q)), \end{aligned} \quad (4)$$

where G_{dat} is the *SSD* of colors of weathered pixels in patches (blue pixels), $G_{neigbor}$ is the *SSD* of colors of overlapping pixels with neighboring patches (orange pixels). $\mathcal{K}_p$ is a rectified weathering degree for which the weathering degree of a "non-weathered" pixel is set as 0 but that of a "weathered" pixel is unchanged. This distance measure $d(I,T)$ prioritizes the appearance similarity in highly-weathered regions, and otherwise maintains visual coherence among neighborhoods. This optimization can generate a natural-looking weathered texture that satisfies not only seamless transition with the already-weathered regions but also conservation of appearance features of the weathering exemplar. In our implementation, we apply coarse-to-fine refinement in two steps while changing sizes of weathering patches. We first search for the patches using 60×60 pixels for preserving global structures of the weathering exemplar, and then update the patches using 20×20

Figure 7: *Various objects at different stages of weathering. For each triplet of images, the left one is the input image, and the others are weathering results.*

pixels to represent more detailed features. The patch search process can be accelerated using PatchMatch [BSFG09] with a modification of the offset for looking up neighboring patches, i.e., from one pixel to the overlapping tile size. In the synthesis process, we use the minimum error boundary cut [EF01] to determine the cut between the neighboring overlapped blocks.

3.4. Weathering effects rendering

Using the global shading and the weathering patches in the previous section, the final image I' is generated. Let I be the input image, Z be the weathered texture that is generated to cover non-weathered regions using weathering patches, and S be the shading map. In this process, the luminance values in Lab color space are multiplied by shading values. Here, we synthesize the weathered texture in the pixels whose weathering degrees D are more than a threshold τ because we assume that the weathering effects start to appear in regions where weathering degrees are above the certain threshold. Following this, the final image is computed as:

$$I'_{lum} = (D'Z_{lum} + (1 - D')I_{lum}) * S, \quad (5)$$

$$I'_{hue} = D'Z_{hue} + (1 - D')I_{hue}, \quad (6)$$

$$D' = \begin{cases} D & \text{if} \quad D > \tau \\ 0 & \text{otherwise.} \end{cases}$$

where $*$ denotes channel-wise multiplication, the terms with subscript *lum* denote the luminance channel and those with *hue* denote chroma channels in Lab space, respectively. We use $\tau = 0.7$ for all results in this paper, except Figure 11(b).

Figure 6 shows a comparison between our weathering synthesis and the existing methods [BSFG09, DSB*12]. In contrast to previous methods that generate visual inconsistency, our method creates visually-coherent textures that preserves detailed features of weathering effects.

3.5. Weathering simulation and editing

We generate weathering effects by updating the weathering degree map automatically or manually. For automatic update, our method can adopt several techniques; smooth expansion of weathered regions by adding a smoothed degree map [BW09, EKMF11], overall update of a weathering degree map based on the time-dependent function [XWT*08], and the particle flow simulation [EKMF10]. In the particle simulation, we can easily use their technique for our task by replacing the amount of flow stains to weathering degrees. In other words, the weathering degree of each pixel increase when particles pass through the pixel. For more free editing, we offer a weathering brush that increases/decreases weathering degrees based on distances from the center of the brush circle. We consider Gaussian distribution in the circle to define the amount of change in the pixels for each time step when the user drags a target position. The users can select the editing methods of the weathering effects according to their purpose. In this paper, we use the smooth expansion method for all results except Figure 8, where we used the weathering brush for editing both weathering and de-weathering in a single object.

(a) User inputs (b) De-/weathering

Figure 8: *Weathering and de-weathering using our brush interface. (a) After specifying the most and least weathered pixels in an input image, (b) the user can edit both weathering and de-weathering of the object simultaneously.*

4. Results

We implemented our prototype system with C++, and ran the program on a PC equipped with a 2.8 GHz CPU and 8 GB of memory. The sizes of input images are about HD resolutions (e.g., Figure 2 is 1000×667 pixels and Figure 3 is 1024×681 pixels). In our experiments, the average computation time for the weathering degrees was approximately 0.1 seconds, the generation of a weathering exemplar was approximately 0.5 seconds, and search of weathering patches was approximately 3 seconds. Note that these processes are computed once and a phase of updating weathering degrees as well as synthesizing weathering patches can be computed instantly (e.g., within 0.01 seconds for each time step).

Weathering and de-weathering. Figure 7 shows results of object weathering using our method. The various types of weathering effects are plausibly generated, such as metal rusting, paint peeling, lichen growing, and flower bloom. Figure 2 demonstrates results of de-weathering and weathering, where bricks appear in the weathering examples whereas plaster surfaces are recovered in the de-weathering example. We generate the de-weathering effects by simply inverting the specified weathered/non-weathered pixels. Figure 8 is an example of weathering and de-weathering, where the user interactively edited the image using our brush interface.

Gradual weathering with multiple weathering exemplars. Our method can also generate gradual weathering with different effects. For this, the user additionally specifies the intermediate weathered regions with rectangles as weathering exemplars. Then, an average weathering degree of each rectangular region is assigned to the corresponding exemplar. When the weathering degree map is updated, the weathering effects are synthesized by blending two weathered textures generated from the exemplars that have nearest degrees to target pixels. Note that the two nearest textures are blended linearly without using a threshold in Equations (5) and (6). Figure 9 shows the result of the gradual weathering with two exemplars. In this case, the paint of the wall first peels and then the plaster peels.

Weathering transfer. We can easily apply weathering transfer by replacing the weathering exemplar with a different exemplar of a reference image. This can be done with simple user inputs, i.e., the user specifies the least/most weathered pixels with scribbles in the reference image. Figure 10 illustrates the result of weathering transfer. The green moss is replaced with yellow lichen naturally.

Figure 9: *Gradual weathering using multiple weathering exemplars. Left to right: an original image with user inputs that include weathered pixels and an intermediate weathering exemplar (yellow rectangle); paint peeling; and plaster peeling.*

(a) Input image (b) Reference image (c) Weathering transfer

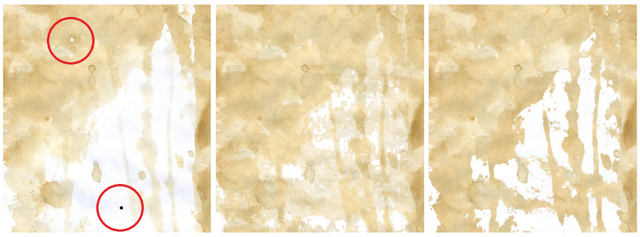
Figure 10: *Weathering transfer. (a) Given an input image, (b) the yellow lichen of the reference image is transferred to the input image. (c) The green lichen is replaced by the yellow lichen.*

4.1. Discussions

Subtle variations of weathering. Our method synthesizes weathering texture in pixels where weathering degrees are above a threshold (Section 3.4). In other words, the threshold determines when weathering effects start to appear and the effects are linearly blended with the original luminance or hue, as in Equations (5) and (6). A relatively-higher threshold works well for weathering effects that change the texture completely (e.g., peeling and mossy growing). For more subtle weathering effects such as stains with small color variations, our method can synthesize natural results by setting the threshold lower. Figure 11 shows weathering effects using different thresholds. Note that we use a fixed threshold $\tau = 0.7$ for all other results in this paper because our target weathering effects with complex variations often change the texture completely, from clean to fully weathered.

Robustness against user inputs. Figure 12 shows a comparison of weathering degree maps with different user inputs. Our method is often robust against the selection of most/least weathered pixels, and it achieves a plausible result with a few inputs thanks to our RBF-based interpolation in the high-dimensional feature space. Even if user inputs affect weathering degree maps sensitively, users can try again easily thanks to the simple user inputs and fast feedback to obtain a desirable result.

Comparisons. Figure 13 shows comparisons with existing methods [BW09, EKMF11]. The existing techniques generate visually-unnatural results because chroma of the each object converges into a single color such as the originally-brown wall. Additionally, detailed features of the weathered regions (e.g., brick patterns or



(a) User inputs (b) $\tau = 0.1$ (c) $\tau = 0.7$

Figure 11: *Weathering synthesis using different thresholds τ in Eqs. (5) and (6). (a) an input image and user inputs; (b) result with threshold $\tau = 0.1$; (c) result with threshold $\tau = 0.7$.*

creeping ivy) are lost. In contrast, our method generates plausible weathering effects that preserve such detailed features of the weathered regions.

Limitations. Our method has several limitations. First, if the input image contains only few weathered regions, we cannot create a coherent weathering exemplar and thus the generated weathering effects will be inconsistent. In this case, we should make an exemplar using another image based on the weathering transfer technique described above. Second, although we demonstrate several de-weathering results, it remains a challenging problem in case that appearance of an object surface drastically changes by weathering effects (Figure 14). In such case, it is difficult to reconstruct the original shape of the object after removing the weathering effects. We also consider that the computation of global shading should be improved to handle more various objects that have complex shapes. Finally, our method currently does not account for the perspective distortion of weathering effects, which we will be able to handle by adding some annotations, e.g., control meshes.

5. Conclusion and future work

In this paper, we have presented an image-based weathering technique for generating weathering effects with complicated feature variation. Our method propagates the detailed features via patch-based global optimization guided by a weathering degree map. The weathering degree map is calculated using Radial Basis Functions for handling materials with wide color variations, unlike previous methods. For generating visually-natural weathering effects, we first create a weathering exemplar including salient weathered features using graph cut, and then synthesize the patches sampled from it while preserving coherency. We have demonstrated our method generates various types of detailed weathering effects interactively.

In future work, we would like to improve the weathering exemplar by sampling patches not only from a rectangular region but also from the whole weathered region to account for a wider variety of weathering features, which will also suppress repetitions of similar image patches. Additionally, we would like to automatically extract multiple exemplars that can represent gradual weathered stages and construct layered structure of different weathering texture, similarly to the layer-based texture synthesis technique [**?**], for richer representation of the weathering process and evolution of boundaries between the weathered and non-weathered regions.

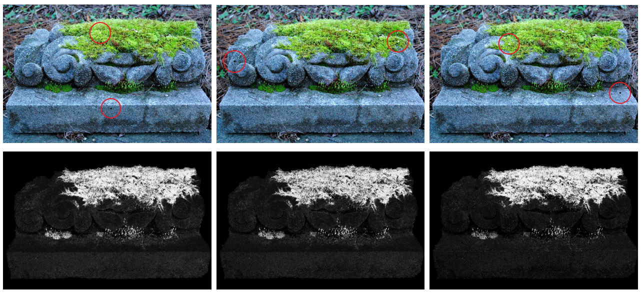


Figure 12: *Comparison of weathering degree maps with different user inputs. The upper row shows user inputs and the lower shows resultant weathering degree maps. Our method is often robust against the selection of most/least weathered pixels.*

Acknowledgements

We thank anonymous reviewers for their suggestions. This work was supported by JST CREST. Yoshihiro Kanamori is funded by JSPS Postdoctoral Fellowships for Research Abroad.

References

[BBO*10] BICKEL B., BÄCHER M., OTADUY M. A., LEE H. R., PFISTER H., GROSS M., MATUSIK W.: Design and fabrication of materials with desired deformation behavior. *ACM Trans. on Graphics (Proc. SIGGRAPH) 29*, 3 (2010). 3

[BJ01] BOYKOV Y., JOLLY M.-P.: Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proceedings of ICCV* (2001), vol. 1, pp. 105–112. 5

[BLR*11] BOSCH C., LAFFONT P.-Y., RUSHMEIER H., DORSEY J., DRETTAKIS G.: Image-guided weathering: A new approach applied to flow phenomena. *ACM Trans. Graph. 30*, 3 (2011), 20:1–20:13. 2

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. (Proc. of SIGGRAPH) 28*, 3 (aug 2009), 24:1–24:11. 2, 4, 5, 6

[BSGF10] BARNES C., SHECHTMAN E., GOLDMAN D. B., FINKELSTEIN A.: The generalized patchmatch correspondence algorithm. In *Proceedings of ECCV 2010* (Berlin, Heidelberg, 2010), Springer-Verlag, pp. 29–43. 2

[BW09] BANDEIRA D., WALTER M.: Synthesis and transfer of time-variant material appearance on images. In *Proceedings of IEEE Computer Graphics and Image Processing (SIBGRAPI)* (2009), pp. 32–39. 2, 4, 6, 7, 9

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of SIGGRAPH '01* (2001), pp. 67–76. 3

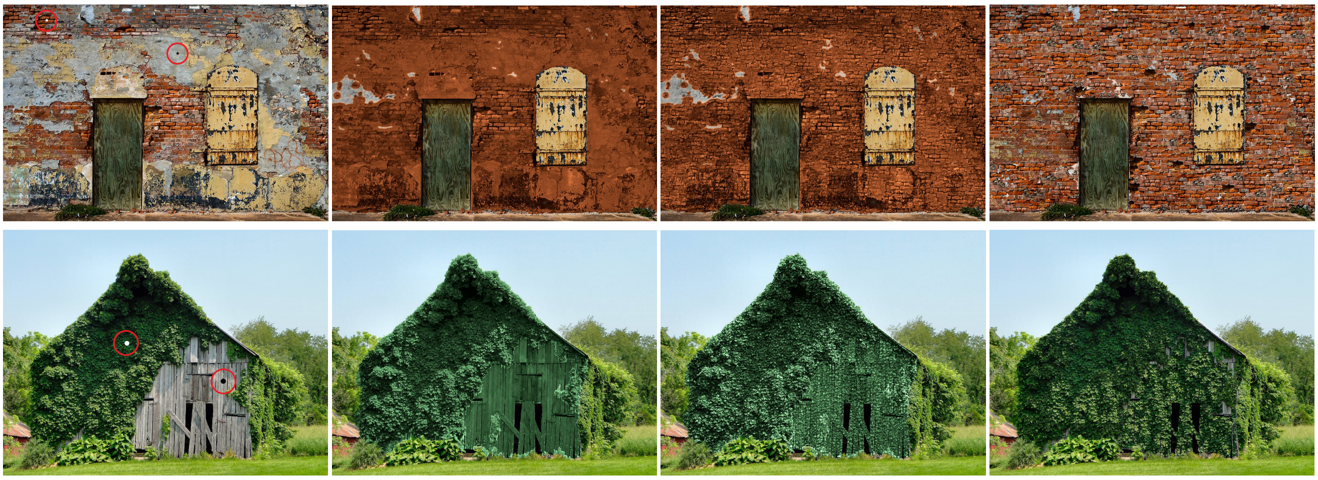
[Cro84] CROW F. C.: Summed-area tables for texture mapping. In *Proceedings of SIGGRAPH '84* (1984), pp. 207–212. 5

[DEJ*99] DORSEY J., EDELMAN A., JENSEN H. W., LEGAKIS J., PEDERSEN H. K.: Modeling and rendering of weathered stone. In *Proceedings of SIGGRAPH '99* (1999), pp. 225–234. 1

[DGA04] DESBENOIT B., GALIN E., AKKOUCHE S.: Simulating and modeling lichen growth. *Computer Graphics Forum 23*, 3 (2004), 341–350. 1

[DH96] DORSEY J., HANRAHANY P.: Modeling and rendering of metallic patinas. In *Proceedings of SIGGRAPH 1996* (1996), pp. 387–396. 1

[DPH96] DORSEY J., PEDERSEN H. K., HANRAHAN P.: Flow and changes in appearance. In *Proceedings of the 23rd Annual Conference on*



(a) User inputs (b) [BW09] (c) [EKMF11] (d) Ours

Figure 13: *Comparisons of image weathering. (a) Given input images and user inputs, (b)(c) the existing methods generate unnatural results, where each chroma of the objects converges into a single chroma and the detailed features such as texture patterns are lost. In contrast, (d) our method generates plausible weathering effects that preserve detailed features of the weathered regions.*

Computer Graphics and Interactive Techniques (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 411–420. 1

[DSB*12] DARABI S., SHECHTMAN E., BARNES C., GOLDMAN D. B., SEN P.: Image Melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2012) 31*, 4 (2012), 82:1–82:10. 2, 4, 5, 6

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *Proceedings of SIGGRAPH '01* (2001), pp. 341–346. 2, 5, 6

[EKMF10] ENDO Y., KANAMORI Y., MITANI J., FUKUI Y.: An interactive design system for water flow stains on outdoor images. In *Proceedings of Smart Graphics 2010* (2010), pp. 160–171. 1, 2, 6

[EKMF11] ENDO Y., KANAMORI Y., MITANI J., FUKUI Y.: Image editing for weathering effects with geometric details for images. In *Proceedings of Computer Graphics International 2011* (2011). 1, 2, 4, 6, 7, 9

[GTR*06] GU J., TU C.-I., RAMAMOORTHI R., BELHUMEUR P., MATUSIK W., NAYAR S.: Time-varying surface appearance: Acquisition, modeling and rendering. *ACM Trans. Graph. 25*, 3 (July 2006), 762–771. 2

[HW95] HSU S.-C., WONG T.-T.: Simulating dust accumulation. *IEEE Computer Graphics and Applications 15*, 1 (1995), 18–22. 1

[HZW*06] HAN J., ZHOU K., WEI L.-Y., GONG M., BAO H., GUO B.: Fast example-based surface texture synthesis via discrete optimization. *The Visual Computer 22*, 9 (2006), 918–925. 2

[KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture optimization for example-based synthesis. *ACM Trans. Graph. 24*, 3 (July 2005), 795–802. 2, 5

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3 (July 2003), 277–286. 2

[LGG*07] LU J., GEORGHIADES A. S., GLASER A., WU H., WEI L.-Y., GUO B., DORSEY J., RUSHMEIER H.: Context-aware textures. *ACM Trans. Graph. 26*, 1 (Jan. 2007). 2

[LH74] LAWSON C. L., HANSON R. J.: Solving least-squares problems. *Prentice-Hall* (1974), 161. 4

[LJH10] LI Y., JU T., HU S.-M.: Instant propagation of sparse edits on images and videos. *Computer Graphics Forum 29*, 7 (2010), 2049–2054. 3, 4

Figure 14: *Typical failure case of de-weathering. Given an input image and user inputs (left), de-weathering fails if the underlying surface details of weathered and non-weathered regions are very different (right).*

[PPD02] PAQUETTE E., POULIN P., DRETTAKIS G.: The simulation of paint cracking and peeling. In *Proceedings of Graphics Interface 2002* (2002), pp. 59–68. 1

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: GrabCut: interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. 23*, 3 (aug 2004), 309–314. 2, 5

[SCSI08] SIMAKOV D., CASPI Y., SHECHTMAN E., IRANI M.: Summarizing visual data using bidirectional similarity. In *Proceedings of CVPR 2008* (2008), pp. 1–8. 2, 4

[SSR*07] SUN B., SUNKAVALLI K., RAMAMOORTHI R., BELHUMEUR P., NAYAR S.: Time-Varying BRDFs. *IEEE Transactions on Visualization and Computer Graphics* (Mar 2007), 1–8. 2

[WSI07] WEXLER Y., SHECHTMAN E., IRANI M.: Space-time completion of video. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 3 (2007), 463–476. 2, 4

[WTL*06] WANG J., TONG X., LIN S., PAN M., WANG C., BAO H., GUO B., SHUM H.-Y.: Appearance manifolds for modeling time-variant appearance of materials. *ACM Trans. Graph. 25*, 3 (July 2006), 754–761. 1, 2, 4

[XDR11] XUE S., DORSEY J., RUSHMEIER H.: Stone weathering in a photograph. *Computer Graphics Forum 30* (2011), 1189–1196. 2

[XWT*08] XUE S., WANG J., TONG X., DAI Q., GUO B.: Image-based material weathering. *Computer Graphics Forum 27* (April 2008), 617–626. 1, 2, 4, 6