Adaptable Aging Factory for Multiple Objects and Colorations

Anonymous

Abstract

The visual effects of synthetic objects and environments from modern video games and movies require an impressive amount of detail to properly duplicate their real world equivalents. Among these details, effects produced by aging are particularly hard to handle, and adding them is significantly time-consuming. Existing methods such as physically-based and empirical simulations are not suitable for artists since they require the manipulation of complex physical parameters, and their results are difficult to control. Our approach offers a framework for quickly adding aging effects based on a *simple example*. By defining an *aging recipe* based on local properties, an artist can easily apply similar effects to different objects or to multiple occurrences of the same object. Also, when aging patterns consist of simple color variations, we propose a color-independent process capable of producing various colorations of the same effect from a single example.

Key words: Realistic rendering, Weathering, Aging effect, Deterioration, Material appearance

1. Introduction

During the last decade, the level of realism seen in video game and movie visual effects has increased significantly. Nevertheless, it is still a very important concern for production studios. In this never-ending challenge, several aspects need to be considered including aging effects. In synthetic environments, objects often seem too perfect when compared to their real world equivalents. Effects such as scratches, bumps, dust, engravings, corrosion, or even patinas must be added to these synthetic objects to make them convincing. Obviously, adding these effects manually is time-consuming, and it is impractical to alter the multitude of objects contained in a given environment. For instance, a classroom scene including multiple chairs and desks that all need to be aged similarly, yet not identically, would require a significant amount of time to alter manually.

Over the years, quite a few methods have been proposed to partially solve this problem, but none is completely appropriate for production studios. Some are hard to control for artists because operating them requires scientific knowledge, while some require complex and costly capture equipment. Furthermore, only a few of these methods allow an artist to quickly generate multiple aged occurrences of the same object, as required in the classroom example. Consequently, in most studios, this kind of work is still done manually. Our goal was therefore to develop an easy-to-use semi-automatic method for adding aging effects using a limited amount of manual work. The key contributions of our work can be summarized as follows:

- A user-friendly semi-automatic framework for quickly adding aging effects based on a simple example.
- A property-based method to intuitively define aging patterns. It can also be used to similarly age objects of different shapes or to easily produce multiple aged occurrences of the same object.
- A color-independent synthesis process used to produce multiple colorations of aging effects from a single example. The process also takes advantage of bump mapping to increase realism without capturing complex BRDFs.

2. Related Work

Since realism of synthetic images is so important for production studios, a lot of work has been done to simulate aging effects on 3D objects. Most of the proposed methods can be categorized in two groups: the methods designed for a specific effect, and those that address many effects. This section presents a quick review of these two groups, highlighting the most important techniques, their advantages and their drawbacks. Refer to Dorsey et al. [1] or Mérillou and Ghazanfarpour [2] for detailed surveys.

Among the methods designed for a specific phenomenon, physically-based simulations have been often proposed to address several aging effects such as water flows on objects [3], stone weathering [3], metal corrosion [4, 5], fractures of object [6], cracks on drying 3D solids [7], etc. These approaches usually produce highly realistic results but require lengthy calculations and unintuitive physical parameters. Furthermore, they are often hard to control when trying to achieve a specific appearance. Alternatively, several empirical methods have been proposed to deal with aging effects such as surface cracks [8], paint peeling [9], surface aging by impacts [10], etc. These approaches offer simplified simulation models to replicate real world phenomena. Consequently, they require less computation, and their parameters are typically more intuitive for artists. Still, the main drawback of both physically-based and empirical

*Corresponding author
[1]Telephone: 1-514-396-8800 Ext.: 7476 Fax: 1-514-396-8405

simulations is that each of their techniques can only replicate a specific effect. As a result, artists need to learn how to work with multiple methods and their set of parameters to properly age their environments.

Lately, researchers have reoriented their work to propose approaches that can handle multiple types of aging effects. Image-based aging methods provide this versatility by using example images as inputs, which is also more intuitive than the traditional physical parameters. Gu et al. [11] propose the capture of the time-varying appearance of an example using an advanced camera setup to reproduce it onto synthetic objects through texture synthesis. Using their setup, they create a database containing multiple effects on multiple materials, including burning wood, decaying fruits and rusting metal. Control over the results is achieved by manually specifying the aging rate and the offset parameters for every point of the surface. Lu et al. [12] propose a similar method, using comparable capture equipment. They introduce the use of local properties to define the position of the aging effects on the object. Wang et al. [13] and Xue et al. [14] capture the aging effect at a single point in time and infer its evolution using what they call an appearance manifold. All these image-based methods do not require calculations as lengthy as those in physically-based approaches, they do not use complex parameters, and they work with a wide variety of aging effects. Moreover, several image-based methods allow the generation of aging patterns on objects at different stages of their evolution in a continuous manner. Their main drawback is the complexity and the costs associated with the capture process, making them unusable by regular production studios.

Finally, even if current image-based methods [11, 12, 13, 14, 15] can handle multiple types of effects, they still need too much manual work from artists to produce similar aging on various objects. In fact, they are not designed to handle the generation of multiple aged occurrences since control over the results is achieved either through target masks painted manually or through local properties, constant on a given object.

Our image-based approach distinguishes itself by using a simplified capture process, by reducing the time needed to adjust the appearance of results, and by taking advantage of normal maps during the lighting calculation to increase the level of realism. Further, it is designed to rapidly generate multiple similar occurrences of the same object through our mask generation tool. Finally, our approach proposes a color-independent synthesis process to produce different colorations of a given aging effect from the same example image.

3. Aging through Constrained Texture Synthesis

Before getting into the details of our framework, it is important to have a good high-level understanding of aging through texture synthesis. Fig. 1 gives a quick overview of a typical image based aging method using constrained texture synthesis [13, 15].

As inputs, image-based aging methods need a capture image, a source effect mask, and a control mask. The capture image contains an example of a desired aging effect, usually obtained from a photograph. The source effect mask is obtained through segmentation of the capture image, and identifies the aged regions on the example. Finally, the control mask is a binary image representing the desired pattern for the aging effect to be added, which is manually painted by an artist. The system then synthesizes new aging effects, based on the capture image, which fit the control mask through a constrained texture synthesis algorithm [16, 17].

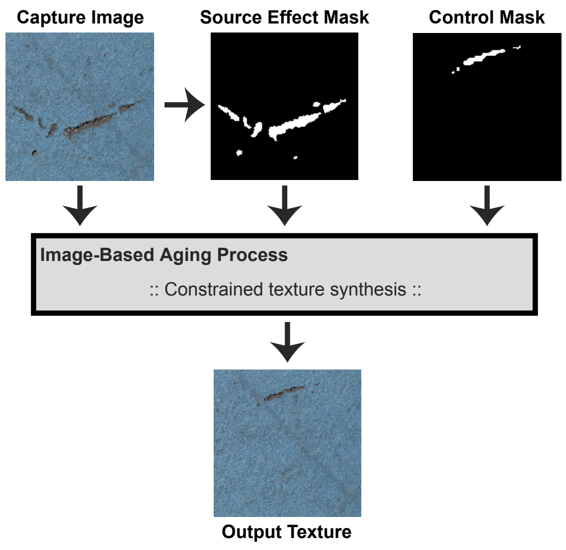

Capture Image
Source Effect Mask
Control Mask
Image-Based Aging Process
:: Constrained texture synthesis ::
Output Texture

Fig. 1: Overview of a typical image-based aging process.

This general approach has two key benefits when compared with other methods. First, the capture process, a plain photograph, is quick, simple, inexpensive, and easy to incorporate into the standard pipeline of a production studio. Also, the control mask provides an adequate and intuitive control for artists to adjust the results. On the other hand, depending on the aging effect, manually creating this mask can be long, tedious, and repetitive, especially if multiple similar masks need to be produced to age an entire environment. This problem will be addressed in Sect. 3.1. Furthermore, since the process synthesizes RGB values based on a specific example, it is sometimes hard to produce multiple colorations of aging effects or to properly transfer the effect to an object of a different color. In some cases, when the color information contained in the texture is essential, it is impossible to envisage otherwise. However, when the color variation within the aged regions is relatively minor, it is conceivable and advantageous to avoid synthesizing RGB values. This will be examined in Sect. 3.2.

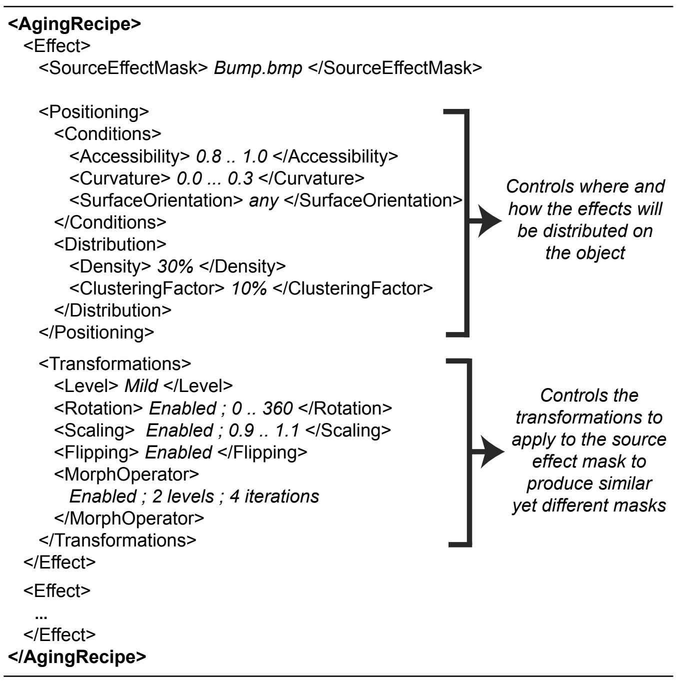
3.1. Automatic Mask Generation Based on Local Properties

Depending on the circumstances, manually painting binary masks to control the appearance of the result can be tedious and time-consuming. Obviously, if an artist wants to add a single patch of rust on a single metallic object, creating the corresponding mask will be quite fast. However, for a severely aged surface with multiple scratches, the mask will not be that easy to produce. According to previous experimentations [15], painting the control masks accounts for more than 90% of the time spent during typical aging through texture synthesis. Furthermore, to put together an environment containing several occurrences of the same object (e.g. multiple desks in a classroom), many different masks need to be created so that each occurrence of the object is similarly aged. The time required to produce such an environment is multiplied by the number of needed occurrences. The same problem stands if the artist wants to age different objects in a similar manner (e.g. a dining table and its chairs). Therefore, the proposed method needs to eliminate this manual painting by providing the artist with a tool to automatically generate masks that are similar to, yet different from the example contained in the capture image. Algorithm 1 presents an overview of the proposed procedure.

Algorithm 1: Pseudo-code for the mask generation tool.

```
Input: AgingRecipe, ObjectProperties
foreach Effect in AgingRecipe do
  // Step 1: Identify texels with appropriate properties
  Candidates ← { }
  foreach pixel P in Effect.SourceEffectMask do
    if ObjectProperties match Effect.Positioning.Conditions then
      Candidates.add(P)
  // Step 2: Generate the control mask
  Density ← 0.0
  while Density < Effect.Positioning.Distribution.Density do
    TransformedMask ← Effect.SourceEffectMask
    processTransformations(TransformedMask,
    Effect.Transformations)
    Pos ← random(Candidates)
    copyEffect(TransformedMask → ControlMask at Pos)
    removeAgedPixelsFrom(Candidates)
    Density ← recalculateDensity()
return ControlMask
```

The key element used as input to this procedure is the *aging recipe*. A typical example in XML format is given in Fig. 2. The first segment of the *aging recipe* allows an artist to define where, and how the effects will be distributed on an object. Algorithm 1 initially processes this information to identify candidate positions by comparing local properties of the given object with acceptable ranges provided in the *aging recipe*. As in the work of Lu et al. [12], several local properties are involved, such as accessibility [18], curvature [19] and surface orientation. These properties allow the artist to define effect positioning in terms of attributes instead of texels, vertices or faces. This way, artists can associate a specific effect to explicit ranges of the local properties in a high-level manner. For instance, an *aging recipe* could associate bumps to highly accessible regions more likely to be smacked and mold to less accessible areas retaining moisture. Therefore, the same *aging recipe* can be used on different objects to produce a similar deterioration. Furthermore, the randomness used with this property-based positioning allows artists to quickly generate multiple occurrences of the same object, which is hard to achieve using previous image-based techniques [11, 12, 13]. The property set considered is not fixed, and can easily be extended to properly characterize and distinguish every region of the object surface. The density parameter controls how many effects will be positioned on a given surface while the clustering factor determines the overall distribution (uniformly scattered or grouped effects). Finally, for absolute control over the effect positioning, the artist can still use manual painting if needed.



```
<AgingRecipe>
  <Effect>
    <SourceEffectMask> Bump.bmp </SourceEffectMask>

    <Positioning>
      <Conditions>
        <Accessibility> 0.8 .. 1.0 </Accessibility>
        <Curvature> 0.0 ... 0.3 </Curvature>
        <SurfaceOrientation> any </SurfaceOrientation>
      </Conditions>
      <Distribution>
        <Density> 30% </Density>
        <ClusteringFactor> 10% </ClusteringFactor>
      </Distribution>
    </Positioning>
    <Transformations>
      <Level> Mild </Level>
      <Rotation> Enabled ; 0 .. 360 </Rotation>
      <Scaling>  Enabled ; 0.9 .. 1.1 </Scaling>
      <Flipping> Enabled </Flipping>
      <MorphOperator>
        Enabled ; 2 levels ; 4 iterations
      </MorphOperator>
    </Transformations>
  </Effect>
  <Effect>
    ...
  </Effect>
</AgingRecipe>
```

Controls where and how the effects will be distributed on the object
Controls the transformations to apply to the source effect mask to produce similar yet different masks

Fig. 2: An example of an aging recipe in XML format.

Using local properties to position aging effects in a high-level manner is an efficient way to reduce or eliminate manual masks painting, which is a problem of some image-based methods [11, 13, 15]. Nevertheless, when considering the problem of several occurrences, they will obviously share the same local properties since they are constant on a given object. Thus, if these properties control the shape of the generated effects, results will be very similar from an occurrence to another. With our method, local properties are only used for positioning and do not influence the final shape of the aging effects as in Lu et al. [12].

With this in mind, the second segment of the *aging recipe* parameterizes the resulting shape of an effect by controlling how Algorithm 1 will alter the source effect mask. Several transformations, such as rotation, scaling and flipping are used. Thus, as a part of an *aging recipe*, the artist selects the appropriate types of transformations and provides acceptable ranges for the corresponding parameters (e.g. rotation angle, scaling factor, etc.). The *level* parameter controls how many transformations will be applied and, thus, specifies how far the control masks will diverge from the source.

Further, a custom morphological algorithm has been developed to alter the appearance even more (see Algorithm 2). The idea is to alter the boundary of the source effect mask with morphological operations. Instead of applying either erosion or dilation to every texel of the boundary, each texel is treated independently. In fact, for each texel, dilation or erosion is randomly selected and performed. This process is repeated for a certain number of iterations. To further change the overall shape of the mask, this process is applied at multiple resolution levels. Obviously, at coarser resolutions, every erosion or dilation changes the shape of the effect significantly. At finer resolutions, alterations are more subtle. Parameters such as the number of levels and the number of iterations can be adjusted to control the deformation intensity. This morphological transformation works pretty well on blob-like discrete effects such as bumps, scratches or rust patches.

Algorithm 2: Pseudo-code for the morphological operator.

```
Input: SourceEffectMask, NbLevel, NbIteration
computeSubsampledPyramid(SourceEffectMask, NbLevel)
ModifiedMask ← SourceEffectMask
foreach Level L, from coarsest to finest do
  for i ← 1 to NbIteration do
    Border ← computeBorder(ModifiedMask(L))
    foreach each pixel P in Border do
      if random(0..1) > 0.5 then
        applyDilation(ModifiedMask(L), P)
      else
        applyErosion(ModifiedMask(L), P)
  // Upsample to next level using nearest neighbor interpolation
  upsample(ModifiedMask(L), ModifiedMask(next L))
return ModifiedMask
```

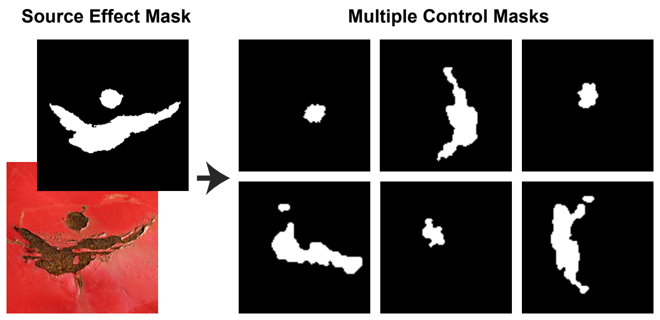


Fig. 3: Multiple control masks produced with our automatic mask generation tool.

All in all, this process alters the source effect and allows the system to automatically produce similar masks. Fig. 3 and Fig. 4 demonstrate the capabilities of this tool. Masks presented in Fig. 3 were generated using all available transformations. In the *aging recipe*, rotation angles were set to vary from 0 to 360 degrees and scaling factors from 90% to 110%. As for our morphological operator, two resolution levels were enabled in order to foster minor changes. As shown in Fig. 4, these effects were positioned on hydrants using accessibility and curvature.

In summary, with this tool, artists can quickly and effortlessly generate multiple control masks from a single example to similarly age various occurrences of the same object, and this, without any manual painting. This tool can also provide artists with masks used to apply different, yet comparable, deterioration to different objects. These automatically generated masks are then

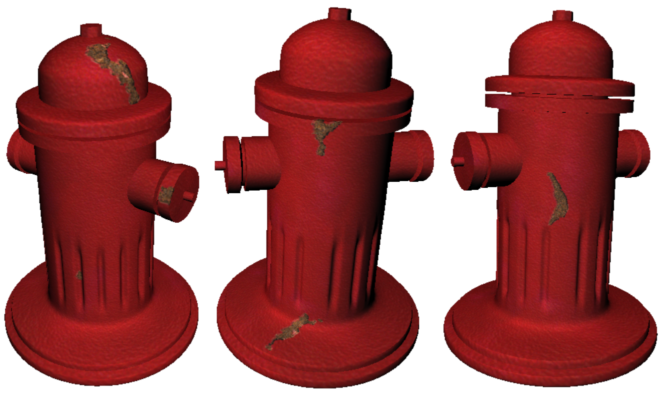


Fig. 4: Control masks applied on multiple occurrences of the same object.

subjected to the synthesis process described in Fig. 1 to produce the corresponding aging effects.

3.2. Color-Independent Synthesis

As introduced earlier, another irritant of the original image-based approach is that it synthesizes RGB values coming from a specific example. Therefore, if the capture image contains a *blue scratch*, it will not be possible to generate a similar *red scratch*. A comparable problem comes up when trying to transfer a scratch captured on a *blue object* onto a *red object*, since the transition from the effect to the base color should be different. Such a need is not uncommon because several objects, such as pieces of furniture, exist in a variety of colors. Fig. 5 shows samples of different-colored tabletops with similar aging effects from antique finish furniture, courtesy of Meubles Canadel.

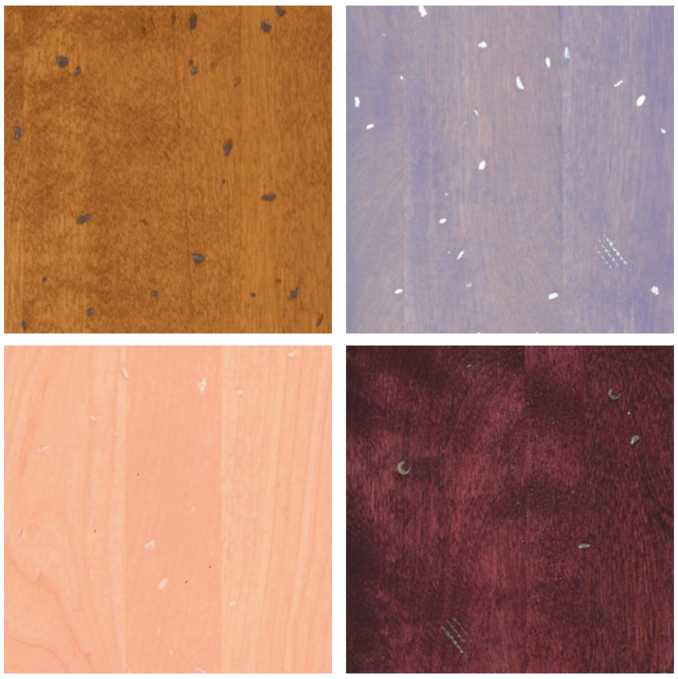


Fig. 5: Similar aging effects on different-colored tabletops, courtesy of Meubles Canadel.

To reproduce such effects, the original synthesis process would require an example for each coloration. The quantity of capture images required to build a complete effect database becomes impractical. This can turn out to be a problem, especially if it is difficult to find a real world example of the desired color. Therefore, artists need to be able to control the output in order to produce a different coloration of aging effects using the *same* example.

For several aging effects, the color inside the affected region is much more correlated to the depth of the effect than to anything else (e.g. bumps, scratches, dust, etc.). To be able to easily generate various colorations, synthesizing depth values will allow a better control on the color of the effect. These grayscale textures will be referred to as *height maps*. After changing the capture image of Fig. 1 with a corresponding height map, a new one matching the control mask will be generated using a typical constrained texture synthesis algorithm [16, 17]. This synthesized height map will finally be interpreted at run-time by a fragment shader in order to determine the color to display. This new synthesis process is illustrated in Fig. 6.

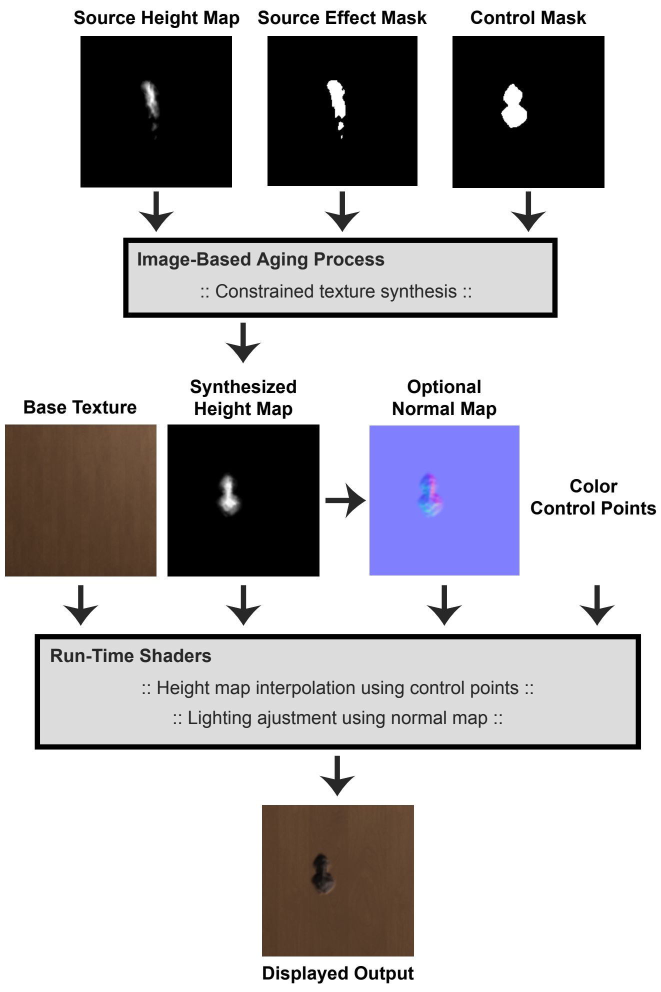
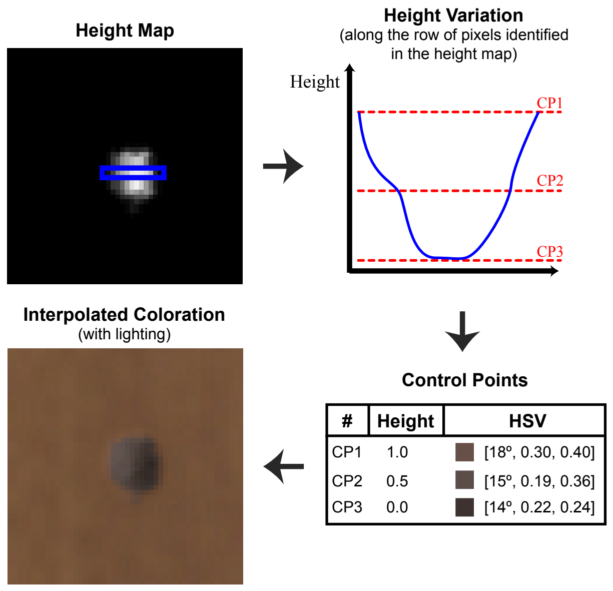

Source Height Map
Source Effect Mask
Control Mask
Image-Based Aging Process
:: Constrained texture synthesis ::
Base Texture
Synthesized Height Map
Optional Normal Map
Color Control Points
Run-Time Shaders
:: Height map interpolation using control points ::
:: Lighting ajustment using normal map ::
Displayed Output

Fig. 6: Overview of our novel color-independent synthesis process.



#	Height	HSV
CP1	1.0	■ [18°, 0.30, 0.40]
CP2	0.5	■ [15°, 0.19, 0.36]
CP3	0.0	■ [14°, 0.22, 0.24]

Fig. 7: Control points calibration and interpolation.

In this context, artists need to be able to control the appearance of the effect in relation to the variation of the corresponding height. To achieve this, a series of color control points are calibrated by the artist to produce the desired variation in appearance. As shown in Fig. 7, a color control point is nothing more than an HSV value paired to a specific height. Therefore, by providing different control points using the same height map, an artist can easily reproduce various colorations from the same example.

For some aging effects and materials, this method tends to oversmooth the appearance of the aged regions. An effective solution to this problem is to apply noise operators to the height maps prior to the run-time interpolation, thus simulating texture grain. In addition, the proposed approach also enables the straightforward generation of bump maps from the synthesized height maps. These bump maps, used in the lighting calculations, are an essential part of the visual cues of many aging effects. This further reinforces the overall realism without the need for a complex BRDF capture setup. Algorithm 3 presents an overview of the fragment shader developed in our implementation.

Algorithm 3: Pseudo-code of the fragment shader.

```
Input: HeightMap, ControlPoints, BaseTexture, TextureCoord
Height ← fetchHeight(HeightMap, TextureCoord)
BaseColor ← fetchColor(BaseTexture, TextureCoord)
if Height is Base Height then
  Color ← BaseColor
else
  CP1, CP2 ← findClosestControlPoints(ControlPoints, Height)
  // Linear interpolation
  Color ← interpolateColor(Height, CP1, CP2)
// Apply lighting in tangent space using bump mapping
Normal ← fetchNormal(HeightMap, TextureCoord)
DisplayedColor ← applyLighting(Color, Normal)
return DisplayedColor
```

With these modifications, the original pipeline goes through a few changes. Initially, the artist must create a source height map corresponding to the desired effect from the RGB capture image. This can be done quickly using common sculpture tools in standard 3D modeling software. If available, this could also be done on a physical example using a depth scanning system [20]. The corresponding source effect mask must also be created. Then, the automatic mask generation tool discussed in Sect. 3.1 is used to generate a control mask. These images are then supplied to the synthesis algorithm to produce a matching height map which will be interpreted at run-time by our fragment shader. As discussed earlier, the artist must adjust the control points to correctly represent the desired color. In terms of coloration, the appearance of the displayed result will be entirely manipulated by these controls points and will have no connection to the RGB capture image. This independence allows artists to produce various colorations without the need of multiple capture images. Fig. 8 shows a few results imitating real life cases given in Fig. 5.

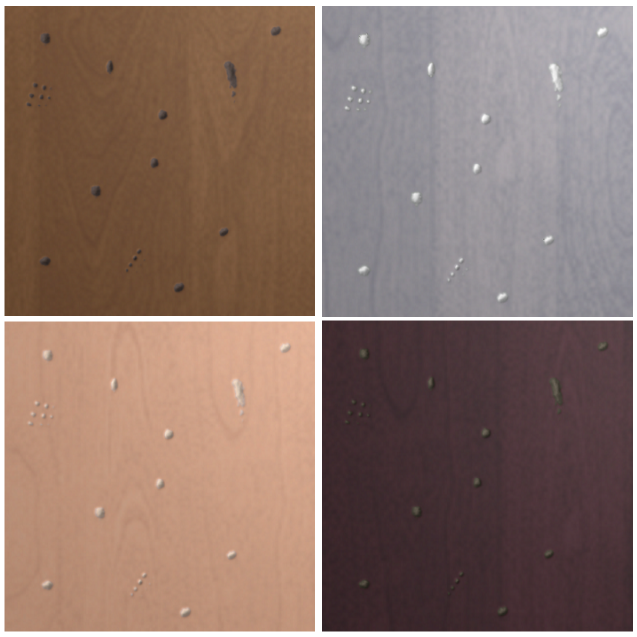


Fig. 8: Results from the color-independent synthesis process.

4. Results

This section presents a few results obtained with our approach. It has been tested with various types of materials, such as wood, concrete, bronze, and stone, and with a variety of aging effects, from scratches to engravings. Fig. 9 replicates patina and mold propagation, respectively on a bronze and a stone sculpture. Obtained results are similar to what is presented in Wang et al. [13], without the use of multiple example images capturing the different colorations. Fig. 10 shows the capability to quickly generate multiple occurrences of a given object and to combine various effects. Fig. 11 illustrates the entire process by presenting multiple occurrences of various colorations. Finally, Fig. 12 compares our results to a real scenario.

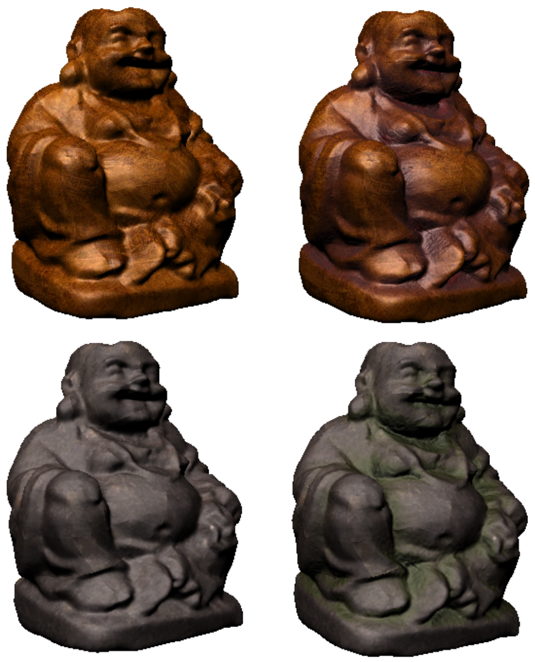


Fig. 9: Patina and mold propagation, respectively on a bronze and a stone sculpture using identical height maps and different control points.

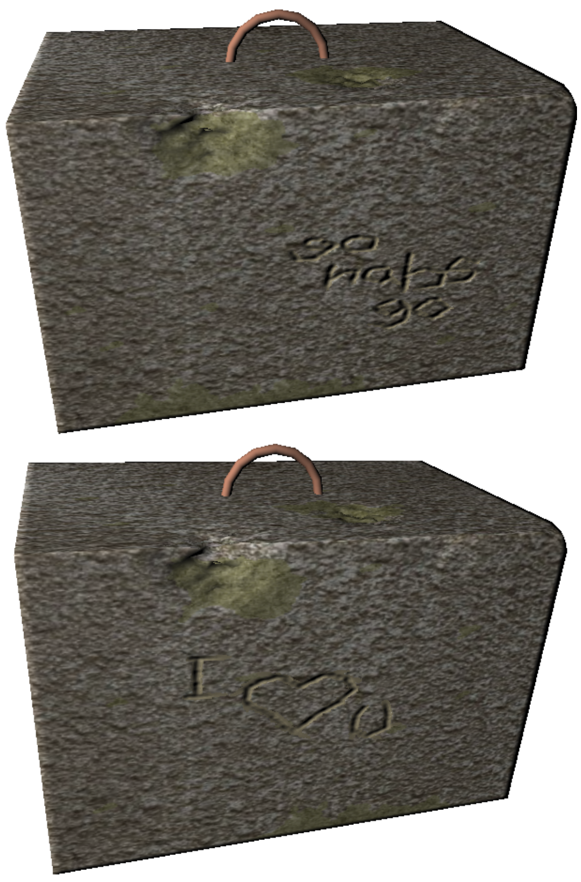


Fig. 10: Mold and engravings on concrete blocks using different height maps and identical control points.

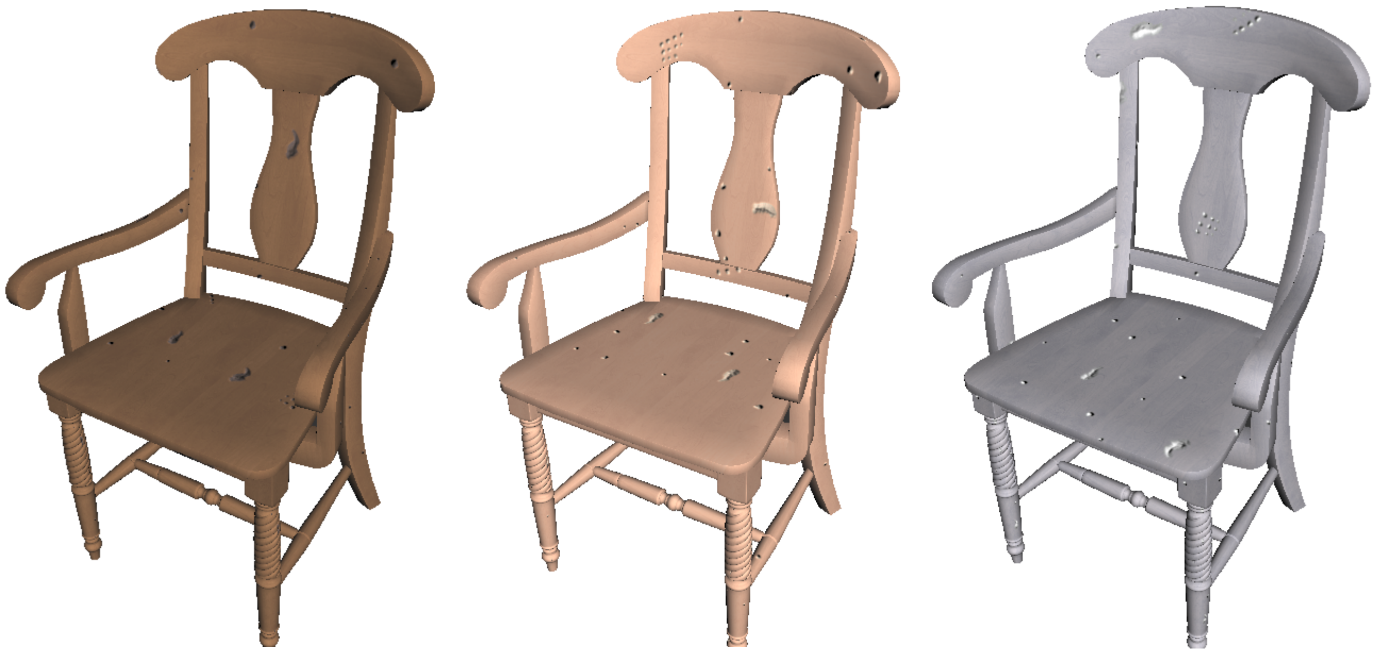


Fig. 11: Aged wooden chairs using different height maps and different control points.

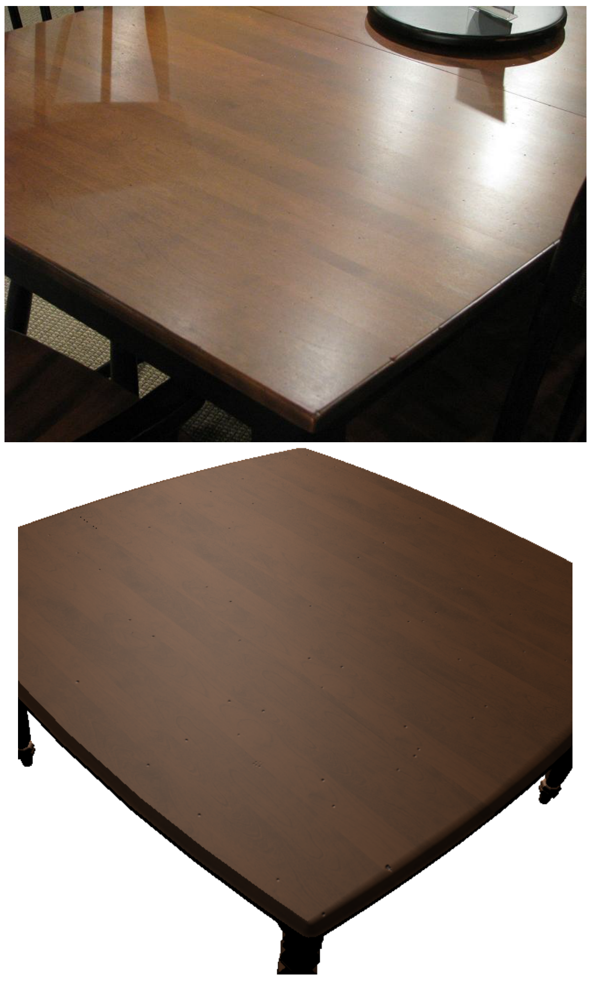


Fig. 12: Picture of a real aged wooden table, courtesy of Meubles Canadel, and its synthesized equivalent.

4.1. Performance

This section presents a performance analysis of the results found in this paper. Table 1 contains important statistics for the mask generation tool discussed in Sect. 3.1 and Table 2 covers the synthesis process explained in Sect. 3.2.

Table 1: Computation times for the mask generation process.

Figure	Texture Size	Number of Effects	Mask Gen. Time
Fig. 3 (1)	1 024 x 1 024	1	3s
Fig. 3 (5)	1 024 x 1 024	1	3s
Fig. 3 (3)	1 024 x 1 024	1	4s
Fig. 3 (4)	1 024 x 1 024	1	10s
Fig. 3 (2)	1 024 x 1 024	1	11s
Fig. 3 (6)	1 024 x 1 024	1	12s
Fig. 4 (1)	1 024 x 1 024	3	13s
Fig. 4 (3)	1 024 x 1 024	3	13s
Fig. 4 (2)	1 024 x 1 024	4	15s
Fig. 8	350 x 350	12	14s
Fig. 10 (1)	1 024 x 1 024	16	18s
Fig. 10 (2)	1 024 x 1 024	18	18s
Fig. 11 (1)	1 024 x 1 024	42	24s
Fig. 11 (3)	1 024 x 1 024	43	24s
Fig. 11 (2)	1 024 x 1 024	50	26s
Fig. 12	2 048 x 2 048	68	48s

As shown in Table 1 and Table 2, using 1 024 by 1 024 textures with typical size aging effects, computation times for both the mask generation tool and the synthesis are quite low. The model used during this process has no effect on the performance since local properties are previously calculated and provided as inputs to the process in 2D textures. A few other variables influence the performance, such as the size of the effects, the amount of transformations applied, etc. They were not included in the

Table 2: Computation times for the synthesis process.

Figure	Aged Pixel	Synthesis Type	Synthesis Time
Fig. 4 (3)	1 463	rgb	3s
Fig. 4 (1)	1 508	rgb	3s
Fig. 4 (2)	2 190	rgb	4s
Fig. 10 (2)	14 216	rgb (mold)	21s
Fig. 10 (1)	15 203	rgb (mold)	22s
Fig. 8	1 492	height	2s
Fig. 10 (1)	1 567	height	2s
Fig. 10 (2)	1 696	height	2s
Fig. 12	2 864	height	4s
Fig. 11 (2)	8 835	height	8s
Fig. 11 (1)	8 900	height	8s
Fig. 11 (3)	9 122	height	8s
Fig. 9	140 708	height	108s

tables for simplicity and because their impact is minimal. These numbers were obtained on a PC with a 2.4GHz CPU, and 3GB of RAM. Such interactive computation times are perfect for the iterative workflow favored in production studios since they enable an artist to quickly adjust the appearance of an object based on reviews and criticisms from peers or artistic directors. Manual painting is limited to special cases, such as text-based effects (see Fig. 10).

The fragment shader used in the color-independent process for run-time interpolation is perfectly suitable for real-time applications. It contains 52 instructions specific to the interpolation, including only one loop. Using 1 024 by 1 024 textures and running on a low-end GPU, a typical aged object is rendered at a frame rate of 60Hz with and without the shader. This frame rate remains constant even in cases where the pixel shader is executed for each and every pixel of the rendered image.

4.2. Limitations

Even if the method is designed to be flexible enough so it can be used in many different situations, some limitations still apply. First, it has the same common restrictions as any other image-based method. It cannot handle aging effects that drastically change the geometry of an object (e.g. fracture, substantial bumps or deformations) since it works only on the surface texture. Also, great care must be taken when capturing the effect example to avoid possible lighting and distortions problems. As for our automatic mask generation tool, it works best with discrete blob-like effects like bumps, and can be harder to use on evolving phenomena like patinas. Finally, our color-independent process is limited to simple color variation within the aged regions. If these regions contain complex patterns, RGB textures must be used for synthesis.

5. Conclusion

In conclusion, we have presented an improved framework used to quickly add aging effects based on a simple example. A built-in property based tool to automatically generate control masks allows artists to easily apply a similar deterioration to different objects and to very quickly generate multiple aged occurrences of the same object. In addition, our color-independent synthesis process can be used to produce multiple colorations of aging effects from a single example. The framework is appropriate for production studios since it does not require specialized hardware, it does not require the manipulation of complex physical parameters, and it provides adequate control over the results. Finally, it is perfectly suitable for real-time applications.

Regarding future work, the framework could be extended to properly handle overlapping effects. For example, it is quite common to observe rust develop on previously scratched points on a piece of metal. In its current state, the framework considers aging effects to be completely independent of one another. This relation could be used as an input to generate a mask for the rust from the mask of the scratches. Also, it could be interesting to experiment with more variables used in the lighting calculation. Obviously, aging effects alter the color of a specific material, but they can also change other characteristics such as its diffuse and specular reflection properties. Such variations could be inferred from height maps or could be added to the synthesis process to improve the results.

Acknowledgements

We would like to thank Meubles Canadel for the funding they provided for this project as well as for their precious collaboration and feedback. We would also like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and the École de technologie supérieure (ÉTS) for supporting this project with scholarships.

References

[1] Dorsey J, Rushmeier H, Sillion F. Digital modeling of material appearance. San Francisco: Morgan Kaufmann; 2007.

[2] Mérillou S, Ghazanfarpour D. A survey of aging and weathering phenomena in computer graphics. Computers & Graphics 2008;32:159-74.

[3] Dorsey J, Hanrahan P. Digital materials and virtual weathering. Scientific American 2000;282:46-53.

[4] Mérillou S, Dischler JM, Ghazanfarpour D. Corrosion: simulating and rendering. In: Graphics Interface 2001 Conference Proceedings, Toronto: Canadian Information Processing Society; 2001, p. 167-74.

[5] Chang YX, Shih ZC. The synthesis of rust in seawater. The Visual Computer 2003;19:50-66.

[6] O'Brien JF, Bargteil AW, Hodgins JK. Graphical modeling and animation of ductile fracture. ACM Transactions on Graphics 2002;21:291-4.

[7] Aoki K, Dong NH, Kaneko T, Kuriyama S. Physically based simulation of cracks on drying 3D solids. In: Computer Graphics International 2004 Proceedings, Washington: IEEE Computer Society; 2004, p. 357-64.

[8] Gobron S, Chiba N. Crack pattern simulation based on 3D surface cellular automata. The Visual Computer 2001;17:287-309.

[9] Paquette E, Poulin P, Drettakis G. The simulation of paint cracking and peeling. In: Graphics Interface 2002 Conference Proceedings, Toronto: Canadian Information Processing Society; 2002, p. 59-68.

[10] Paquette E, Poulin P, Drettakis G. Surface aging by impacts. In: Graphics Interface 2001 Conference Proceedings, Toronto: Canadian Information Processing Society; 2001, p. 175-82.

[11] Gu J, Tu CI, Ramamoorthi R, Belhumeur P, Matusik W, Nayar S. Time-varying surface appearance: acquisition, modeling and rendering. ACM Transactions on Graphics 2006;25:762-71.

[12] Lu J, Georghiades AS, Glaser A, Wu H, Wei LY, Guo B, et al. Context-aware textures. ACM Transactions on Graphics 2007;26:3.

[13] Wang J, Tong X, Lin S, Pan M, Wang C, Bao H, et al. Appearance manifolds for modeling time-variant appearance of materials. ACM Transactions on Graphics 2006;25:754-61.

[14] Xue S, Wang J, Tong X, Dai Q, Guo B. Image-based material weathering. Computer Graphics Forum 2008;27:617-26.

[15] Clément O, Benoit J, Paquette E. Efficient editing of aged object textures. In: Afrigraph '07: Proceedings of the 5th international conference on computer graphics, virtual reality, visualisation and interaction in Africa, New York: ACM; 2007, p. 151-8.

[16] Efros AA, Leung TK. Texture synthesis by non-parametric sampling. In: Proceedings of the international conference on computer vision - volume 2, Washington: IEEE Computer Society; 1999, p. 1033-8.

[17] Hertzmann A, Jacobs CE, Oliver N, Curless B, Salesin DH. Image analogies. In: Proceedings of the 28th annual conference on computer graphics and interactive techniques, New York: ACM; 2001, p. 327-40.

[18] Kontkanen J, Laine S. Ambient occlusion fields. In: I3D '05: Proceedings of the 2005 symposium on interactive 3D graphics and games, New York: ACM; 2005, p. 41-8.

[19] Meyer M, Desbrun M, Schröder P, Barr AH. Discrete differential-geometry operators for triangulated 2-manifolds. Visualization and Mathematics 2003;3:35-57.

[20] Beraldin JA, Cournoyer L, Rioux M, Blais F, El-Hakim SF, Godin G. Object model creation from multiple range images: acquisition, calibration, model building and verification. In: NRC '97: Proceedings of the international conference on recent advances in 3-D digital imaging and modeling, Washington: IEEE Computer Society; 1997, p. 326-33.