Creating Various Styles of Animations Using Example-Based Filtering

Ryota Hashimoto Henry Johan Tomoyuki Nishita

Department of Computer Science
The University of Tokyo
{hashiryo, henry, nis}@is.s.u-tokyo.ac.jp

Abstract

A number of Non-Photorealistic Rendering methods for producing artistic style images have been developed. Recently, a method called "Image Analogies" was proposed. This method is based on the idea of example-based filtering that uses a pair of images (an original image and a filtered image) as training data and creates an image which has a style that resembles the filtered image of the training data. This method has great flexibility since the users can create various styles of images by simply changing the training data. In this paper, we extend "Image Analogies" to create various styles of animations. The coherence between frames is considered by computing the pixel flows between the frames. From experimental results, the proposed method can create nice animation sequences.

1 Introduction

Non-Photorealistic Rendering (NPR) has gained much attention since it can create more impressive images than photorealistic rendering does. The advantage of NPR is that it can omit fine details and produce images which contain only the important portions of the scenes. One of the important fields in NPR deals with the problem of how to produce artistic style images, such as pen-and-ink illustrations, oil paintings, watercolor paintings, and so on. A number of NPR methods for these purposes are proposed and users can choose the most effective rendering method for expressing their idea.

Hertzmann *et al.* [8] proposed a new image synthesis method based on the concept of machine learning. Their method learns a filter from given examples images (which consist of two images, unfiltered and filtered) and applies the learned filter to the target image. The advantage of this method is that it can create various styles of images by simply changing the training data. In the rest of this paper, we will use the term "Image Analogies" to refer to this method.

"Image Analogies", however, cannot directly be applied to animations. In the case of animation, frame-to-frame coherence is very crucial. Simply applying "Image Analogies" to each frame of the animation independently will produce an animation that appears to flicker considerably. In this paper, we extend "Image Analogies" to deal with animation. To maintain the coherence, the pixel flows between the previous frame and the current frame are first computed. Then, based on this information, the method decides whether the pixel values in the current frame should be recomputed or simply copied from the previous frame. As a result, coherency between frames is preserved considerably.

This paper is organized as follows. Section 2 discusses related work on artistic style rendering. Section 3 briefly describes "Image Analogies". Section 4 describes the proposed method for creating various styles of animations. Section 5 shows the experimental results of our method. Section 6 concludes this paper and describes a challenge for future research.

2 Related work

Many methods for creating artistic style images have been proposed. Haeberli [4] created impressionistic images by drawing an ordered collection of brush strokes. Methods for creating pen-and-ink style images have been presented by Salisbury *et al.* [14] and Winkenbach and Salesin [16]. Curtis *et al.* [3] proposed a watercolor painting method using fluid simulation. Hertzmann [6] used cubic B-splines to simulate strokes of various sizes in order to produce stroke-based paintings (*e.g.* oil paintings). Kowalski *et al.* [11] presented an algorithm that uses strokes to render 3D scenes in a stylized manner suggesting the complexity of the scene, such as fur, grass and trees, without representing it explicitly. These previous works can be used only to create specific art-style images. On the other hand, Hertzmann *et al.* [8] proposed an image synthesis method that learns the filter represented by training images and applies it to a target image. As a result, various styles of artistic images can be created by simply changing the training images.

In the case of NPR animation, Meier [13] proposed a method for rendering animations in a painterly style that provides the coherence between frames in animations by modeling surfaces as 3D particle sets. Techniques for maintaining coherence when creating impressionist style animations using the optical flow approach were presented by Litwinowicz [12] and Hertzmann and Perlin [7]. Kaplan *et al.* [9] presented an algorithm for rendering coherent scenes and highly coherent animations in a variety of artistic styles using an editable particle system. Haga *et al.* [5] proposed a method for rendering pen-and-ink-style animations with frame-to-frame coherence by using the stroke information of the previous frame. However, these methods can only create animations of a specific artistic style. Klein *et al.* [10] proposed a method that treats the input video as a space time volume image of data. The coherence between frames is established with the help of the users. Nevertheless, this method creates a flickering animation for certain artistic styles.

3 Overview of "Image Analogies"

"Image Analogies" is based on the recent advancement in texture synthesis algorithms [15, 2]. Texture synthesis aims to create textures with arbitrary size and shapes, which look like the original input texture. The idea of these approaches is to copy the value of the pixel in the original texture which has a neighborhood resembling the neighborhood of the computed pixel in the texture being synthesized.

As input, "Image Analogies" takes three images, the *unfiltered source image* A, the *filtered source image* A', and the *unfiltered target image* B. As output, the method produces the *filtered target image* B', such that

$$A : A' :: B : B'.$$

In other words, the goal is to create image B' that relates to B "in the same way" as A' relates to A.

The approach assumes that the colors at and around any given pixel q in A correspond to the colors at and around the same pixel q in A'. Through out the paper, the symbol q is used to specify both a pixel in A and its corresponding pixel in A'. Similarly, symbol p is used to specify the corresponding pixels in B and B'.

The image synthesis process using a multiresolution approach is as follows:

1. Create image B' from the coarsest to the finest levels.

2. Determine the pixel value of each pixel p in B'_l (image at the l-th level resolution) in a raster scan ordering by finding a pixel q in A'_l which has a neighborhood nearest to the neighborhood of p.

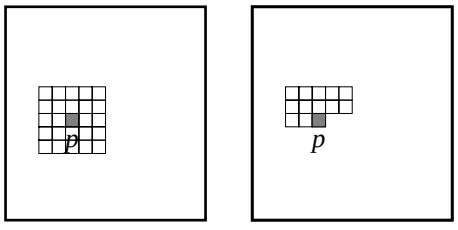


Figure 1. The single resolution neighborhood of pixel p is defined as the union of the L-shaped region which centers on p in the filtered image and the rectangular region which centers on p in the corresponding unfiltered image.

The core of this algorithm is the neighborhood matching. The single resolution neighborhood of pixel p is shown in Figure 1. Intuitively, the size of the neighborhood should be at least as large as the largest structure of the image filter represented by the training data A and A'. However, for images containing large scale structures, a large neighborhood must be used, resulting in increased computation time.

This problem is solved by using a multiresolution synthesis approach. Computational cost is reduced since large scale structures can be represented more compactly with a few pixels at a coarser level resolution. Assume that p' is the corresponding pixel of p in the one level coarser resolution filtered image. The multiresolution neighborhood of p is extended from the single resolution neighborhood of p to include the single resolution neighborhood of p'.

The search in step 2 is done by performing a global search and a local search, and the best search result is returned. The global search is performed using a similar approach to that proposed by Wei and Levoy [15]. The Approximate Nearest Neighbor (ANN) [1] method is used to accelerate the search. The local search uses the approach proposed by Ashikhmin [2] which uses the search result of the preceding pixels and their relative positions.

4 Creating animations

In this section, we first describe an outline of the algorithm, then explain its details.

4.1 Outline

Our algorithm takes two images, the *unfiltered source image* A and the *filtered source image* A', and a sequence of images (animation), the *unfiltered target ani-*

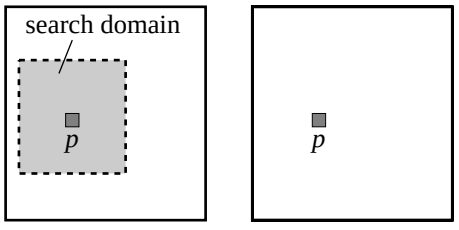


Figure 2. Search domain for motion estimation.

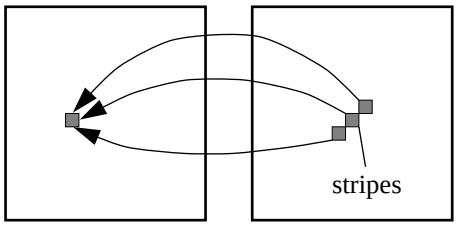


Figure 3. Stripes appear when several pixels in the current frame are set to correspond to a single pixel in the previous frame.

mation $B_1 \cdots B_T$ (where T denotes the length of the animation). The algorithm produces the *filtered animation* $B'_1 \cdots B'_T$ as output, such that

$$A : A' :: B_t : B'_t$$

for every t. In other words, our algorithm creates an animation such that the relationship between B_t and B'_t is same as the relationship between A and A' for every t.

The problem when creating NPR animations is the loss of coherence between frames. Usually, we are aware of the loss of coherence when the animation appears to flicker. The noise that generates the flickering is easily introduced especially when we create animations containing certain artistic styles. In our approach, we try to preserve coherency by maintaining the temporal redundancy of the original animation.

The outline of the proposed method is as follows:

1. Create the first frame B'_1 using the algorithm described in Section 3.
2. Create frames for $t = 2, \cdots, T$ from the coarsest to the finest level using the following steps.
3. Calculate the motion between $B'_{t-1,l}$ and $B'_{t,l}$, (images at the l-th level resolution) by computing the correspondence map $prev$ which maps the pixels in $B'_{t,l}$ to the pixels in $B'_{t-1,l}$ (refer to Section 4.2) .
4. Determine the pixel value of each pixel p in $B'_{t,l}$ in a raster scan ordering by first selecting the search method according to the change between the neighborhood of p and the neighborhood of $prev(p)$ in the filtered image, and then find the nearest pixel q from A (refer to Section 4.3).

4.2 Motion estimation

In motion estimation, for each pixel p in $B_{t,l}$ we search for its corresponding pixel $prev(p)$ in $B_{t-1,l}$. Before we describe the algorithm, we first define the notion of *neighborhood* used in motion estimation. The definition of neighborhood used here is different from that used in "Image Analogies". For motion estimation, the neighborhood of p is defined as an $n \times n$ pixel rectangle with p as its center (n is an odd number). In our experiment, we set n to 5. This value is determined based on experimental results.

The corresponding pixel of p at the previous frame $prev(p)$ is computed by finding a pixel in the previous frame whose neighborhood resembles the neighborhood of p. The similarity between two neighborhoods is measured by the Mean Absolute Difference of the colors of their pixels. In most cases, the motion is relatively small, thus a local search around the same location as p in the previous frame is sufficient to find the corresponding pixel. Based on several experimental results, in practice, we perform the search on pixels in $B_{t-1,l}$ which are located inside an $(6n+1) \times (6n+1)$ pixel rectangle whose center coincides with the location of p (see Figure 2). This approach works well for preserving coherence when we create an animation using "Image Analogies" because it resembles "Image Analogies" in the way that both methods perform searching by comparing neighborhoods of pixels.

However, due to the simplicity of the algorithm, there is a chance that several pixels in $B_{t,l}$ are set to correspond to a single pixel u in $B_{t-1,l}$. This can cause problems since there is a possibility that the value of u is used as the value of those pixels. As a result, stripes (see Figure 3), consisting of pixels of the same color, may appear in $B_{t,l}$. In this case, we employ a heuristic approach which selects the pixel in $B_{t,l}$ with the minimum moving distance as the pixel that corresponds to u. For the rest of the pixels, we assume that they do not have corresponding pixels in $B_{t-1,l}$. From experimental results, this simple solution works well.

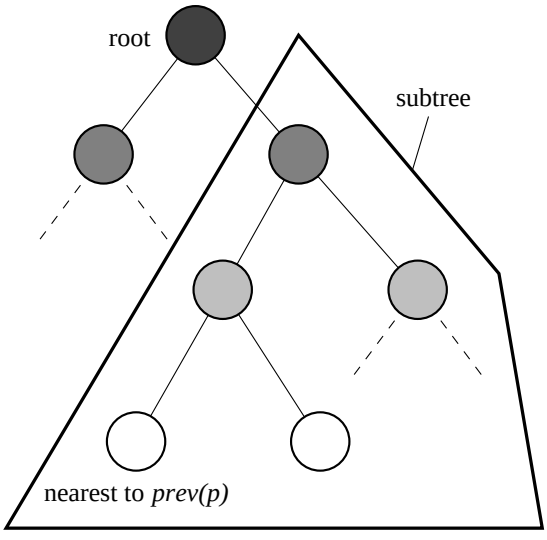


Figure 4. Shortcut global search is performed in the subtree of the ANN tree.

4.3 Coherence preserving search

Using the correspondence map *prev* described in Section 4.2, we measure the coherency degree of pixel p, *coh(p)*, which represents the amount of change in the neighborhood pixels of p in the filtered image from the previous frame. Specifically, $coh(p)$ is defined as the value of the number of pixels in the neighborhood of p for which the colors and relative positions with respect to p are unchanged since the previous frame divided by the total number of neighborhood pixels ($0.0 \leq coh(p) \leq 1.0$). For pixels which do not have corresponding pixels in the previous frame, we assign zero to their coherency values.

According to $coh(p)$, we modify the search methods as follows.

$0 \leq coh(p) < \epsilon_1$
: The neighborhood of p has changed greatly, therefore, the search is performed using both a global and a local search.

$\epsilon_1 \leq coh(p) < \epsilon_2$
: The neighborhood of p has been well preserved, so we assume that the global search result is located near the result of the previous frame. Thus, we can search for the nearest pixel of p with a shortcut global search and a local search. The shortcut global search is performed not in the whole ANN tree, but in the subtree which includes the nearest pixel of $prev(p)$ (see Figure 4). The depth of the subtree is determined in proportion to the value of $coh(p)$.

Table 1. The computation times (minutes per frame) for generating the result animations.

Style of the animation	Computation time
Texture-by-numbers	2
Oil painting	2
Watercolor painting	2
Pen-and-ink	2

$\epsilon_2 \leq coh(p)$
: The neighborhood of p has been very well preserved, so there is no need to search for the nearest pixel of p. Instead, we just set the nearest pixel of $prev(p)$ to be the nearest pixel of p.

The optimal values for ϵ_1 and ϵ_2 are determined based on the results of several experiments. In practice, we set ϵ_1 to 0.3 and ϵ_2 to 0.8, respectively.

5 Results

All of the results presented here (Figures 6 and 8 (see Color Plate for Figure 8), from top to bottom) show several frames taken from the created animations[1].

Figure 5 shows the training images for creating the animations of a flower field. Figure 6 shows the original animation, the resulting animation using "Image Analogies", and the resulting animation using the proposed method. The proposed method outperformed "Image Analogies" with respect to the quality of the created animations. Creating the frames independently using "Image Analogies" produced an animation that appeared to flicker considerably, for example inside the encircled areas of the middle column in Figure 6. On the contrary, the proposed method succeeded to preserve the coherence between frames, for example inside the encircled areas of the right column in Figure 6.

The input movie for the examples shown in Figure 8 was taken using a video camera. The movie was processed in three different styles using the training data shown in Figure 7. The results of applying oil painting, watercolor painting, and pen-and-ink styles to the original movie can be seen in Figure 8. We also used "Image Analogies" to create animations for these examples, however, the resulting animations exhibited severe flickering due to the lack of coherence between frames.

Table 1 shows the length of time required to create one frame of each of the animations using a Pentium 4 3.06 GHz machine. The size of each frame of the animations

[1]Created animations can be seen at the following URL.
http://nis-lab.is.s.u-tokyo.ac.jp/cgi2003_hashimoto/

in Figures 6 and 8 are 320×240 pixels and 352×240 pixels, respectively. The computation time depends on the frame sizes, the sizes of the training images, and the amount of object movement in the animation.

6 Conclusion and future work

This paper has proposed a method that creates various styles of animations using example-based filtering. When creating the animation, the proposed method measures the coherence between frames and uses this information to select the best suited search method. As a result, the amount of observed flickering in the created animations can be considerably reduced. We have shown that the proposed method can be used to create various styles of animations.

Similar to "Image Analogies", the proposed method has a high computational cost. This becomes a problem when we want to create a long animation with frames of a large size. Therefore, we are interested in speeding up the image synthesis process.

Acknowledgment

We would like to thank Toshiyuki Haga for his help and discussion during the early stages of this work. We also would like to thank Hajime Matsui for providing the oil painting image in Figure 7 and Tomohiro Nishita for providing the pen-and-ink image in Figure 7.

References

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions", *Journal of the ACM*, Vol. 45, No. 6, 1998, pp. 891-923.

[2] M. Ashikhmin, "Synthesizing Natural Textures", *Proceedings on 2001 Symposium on Interactive 3D Graphics*, 2001, pp. 217-226.

[3] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin, "Computer-Generated Watercolor", *Proceedings of ACM SIGGRAPH 97*, 1997, pp. 421-430.

[4] P. E. Haeberli, "Paint by Numbers: Abstract Image Representations", *Computer Graphics (Proceedings of ACM SIGGRAPH 90)*, Vol. 24, No. 4, 1990, pp. 207-214.

[5] T. Haga, H. Johan, and T. Nishita, "Animation Method for Pen-and-Ink Illustrations Using Stroke Coherency", *Proceedings of CAD/Graphics 2001*, 2001, pp. 333-343.

[6] A. Hertzmann, "Painterly Rendering with Curved Brush Strokes of Multiple Sizes", *Proceedings of ACM SIGGRAPH 98*, 1998, pp. 453-460.

[7] A. Hertzmann and K. Perlin, "Painterly Rendering for Video and Interaction", *Proceedings of NPAR 2000*, 2000, pp. 7-12.

[8] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image Analogies", *Proceedings of ACM SIGGRAPH 2001*, 2001, pp. 327-340.

[9] M. Kaplan, B. Gooch, and E. Cohen, "Interactive Artistic Rendering", *Proceedings of NPAR 2000*, 2000, pp. 67-74.

[10] A. W. Klein, P. J. Sloan, A. Finkelstein, and M. F. Cohen, "Stylized Video Cubes", *Proceedings of 2002 Symposium on Computer Animation*, 2002, pp.15-22.

[11] M. A. Kowalski, L. Markosian, J. D. Northrup, L. Bourdev, R. Barzel, L. S. Holden, and J. F. Hughes, "Art-Based Rendering of Fur, Grass, and Trees", *Proceedings of ACM SIGGRAPH 99*, 1999, pp. 433-438.

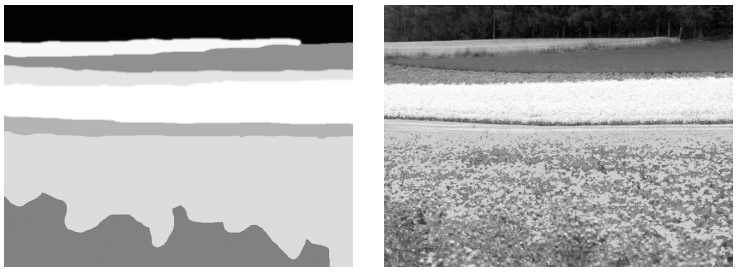
[12] P. Litwinowicz, "Processing Images and Video for an Impressionist Effect", *Proceedings of ACM SIGGRAPH 97*, 1997, pp. 407-414.

[13] B. J. Meier, "Painterly Rendering for Animation", *Proceedings of ACM SIGGRAPH 96*, 1996, pp. 477-484.

[14] M. P. Salisbury, S. E. Anderson, R. Barzel, and D. H. Salesin, "Interactive Pen-and-Ink Illustration", *Proceedings of ACM SIGGRAPH 94*, 1994, pp. 101-108.

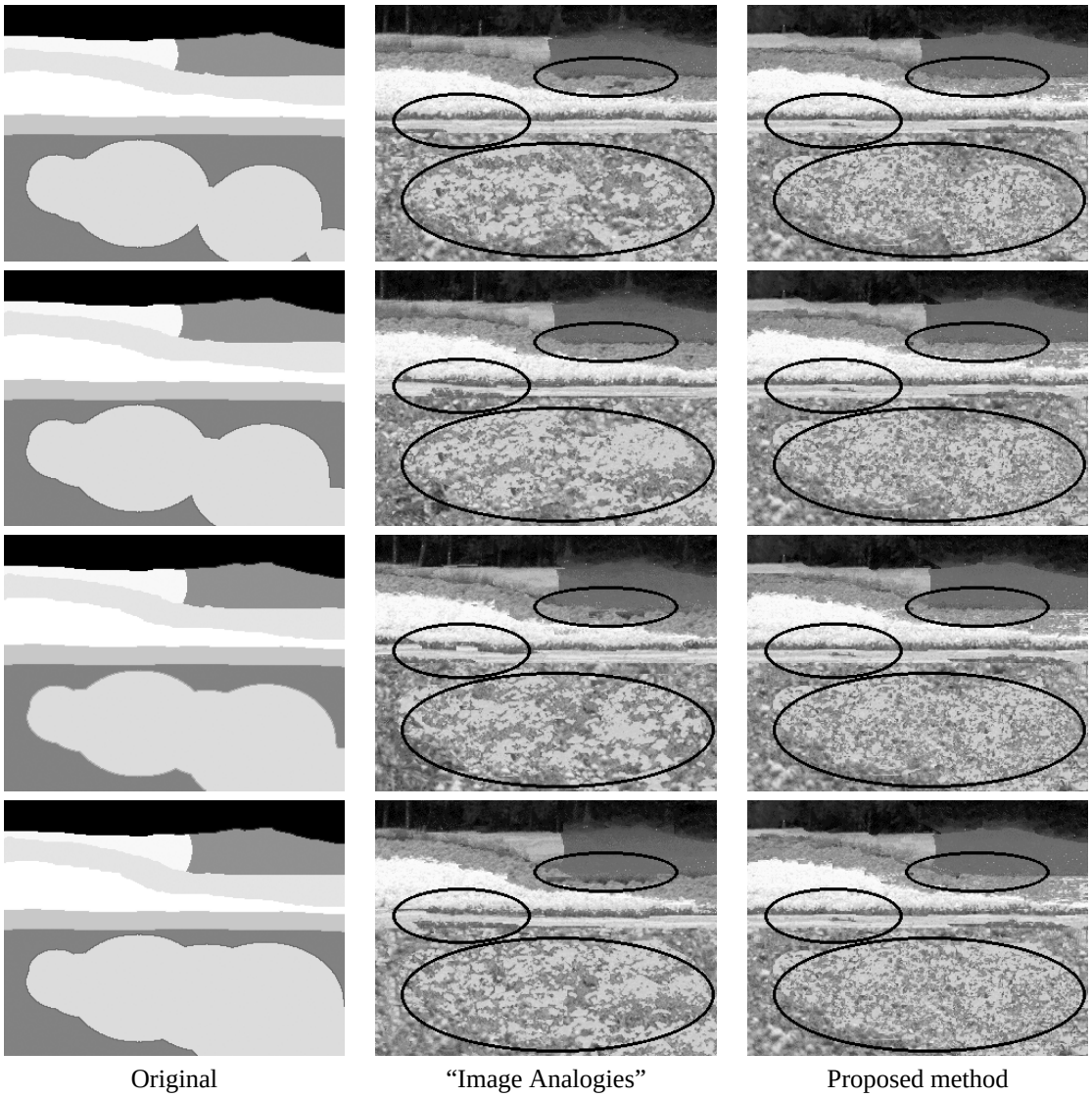
[15] L. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization", *Proceedings of ACM SIGGRAPH 2000*, 2000, pp. 479-488.

[16] G. Winkenbach and D. H. Salesin, "Computer-Generated Pen-and-Ink Illustration", *Proceedings of ACM SIGGRAPH 94*, 1994, pp. 91-100.



unfiltered filtered

Figure 5. The unfiltered and filtered source images for creating animations of a flower field.



Original "Image Analogies" Proposed method

Figure 6. Creating animations of a flower field using a texture-by-numbers approach. The regions inside the encircled areas in the resulting animation using "Image Analogies" flicker considerably while the corresponding regions using the proposed method do not.

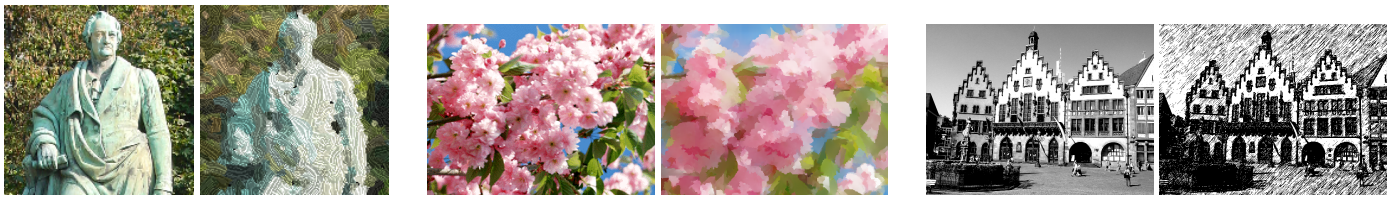


Figure 7. The unfiltered and filtered source images for creating animations in artistic styles.

Figure 8. Original movie and oil painting, watercolor painting, and pen-and-ink versions.