Journal of Zhejiang University SCIENCE A
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Using texture synthesis in fractal pattern design

WEI Bao-gang (魏宝刚)†1, LI Jian-ping (李建平)2, PANG Xiang-bin (庞向斌)1

(1College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)

(2Department of Biosystem Engineering, Zhejiang University, Hangzhou 310029, China)

†E-mail: wbg@zju.edu.cn

Received Oct. 12, 2005; revision accepted Dec. 17, 2005

Abstract: Traditional fractal pattern design has some disadvantages such as inability to effectively reflect the characteristics of real scenery and texture. We propose a novel pattern design technique combining fractal geometry and image texture synthesis to solve these problems. We have improved Wei and Levoy (2000)'s texture synthesis algorithm by first using two-dimensional autocorrelation function to analyze the structure and distribution of textures, and then determining the size of L neighborhood. Several special fractal sets were adopted and HSL (Hue, Saturation, and Light) color space was chosen. The fractal structure was used to manipulate the texture synthesis in HSL color space where the pattern's color can be adjusted conveniently. Experiments showed that patterns with different styles and different color characteristics can be more efficiently generated using the new technique.

Key words: Fractal geometry, Texture synthesis, Two-dimensional autocorrelation function, Pattern design, Image render
doi:10.1631/jzus.2006.A0289 **Document code:** A **CLC number:** TP31

INTRODUCTION

In the last two decades, research on fractal geometry has received much attention for its theoretical significance and practical applications. Pattern design based on fractal geometry is an important application area of fractal techniques in the textile industry (Wang, 2003).

Computer aided pattern design involves the design of textures and colors in the pattern. The source of textures is rich. All kinds of objects and scenery including man-made abstract objects can be used as texture fodder for pattern design, although colour design is mainly dependent on human intervention with respect to filling and processing.

Texture synthesis is a hotspot of research in computer aided geometry design, computer vision, and image processing communities, and finds wider and wider applications such as image analogy (Song *et al*., 2004; Daniilia *et al*., 2000), image inpainting and restoration (Bertalmio *et al*., 2003; Barni *et al*., 2000). Through texture synthesis, various sizes of similar textures can be generated. This characteristic makes it possible to introduce natural or artificial texture patterns into the ones to be devised.

In this paper, we propose a novel fractal pattern design technique based on texture rendering, aimed at making the above idea become reality through the integration of texture synthesis and fractal geometry, thus providing a novel avenue for computer aided pattern design.

RELATED TEXTURE SYNTHESIS AND ANALYSIS

Texture and texture synthesis

Texture can provide the observer with the same sensation and can be depicted using various quantities and types of basic elements, and their organization and arrangement in space. The spatial organization of

*Project supported by the Natural Science Foundation of Zhejiang Province (No. M603228), Zhejiang Science and Technology Plan Project, and Ningbo Science Foundation for Doctor, China

textures can be stochastic, or composed of neighbouring basic elements in pairs or simultaneously-associating several elements. Such association can be structured, probabilistic, or functional.

Through texture synthesis, arbitrarily sized textures can be generated, and the feeling that the patterns occur repeatedly can be eliminated. Texture synthesis based on samples is a hotspot of research on it. At present, there are two kinds of texture synthesis techniques: point matching approach (Efros and Leung, 1999; Wei and Levoy, 2000) and block-by-block approach (Efros and Freeman, 2001; Cohen *et al*., 2003; Kwatra *et al*., 2003). The synthesis method proposed in this paper combines the Wei and Levoy (2000)'s point matching algorithm with the technique that recognizes the size and spatial organization of basic texture elements.

Statistical analysis of textures

A key texture synthesis question is how to generate arbitrarily sized textures using a limited sized texture example. In the point matching algorithms, the region's size of extracted dots plays an important role in the synthesis result. The more approximate the theoretical texture size is to the actual texture size, the better the synthesis algorithm obtained will be. We regard textures as arrangements of similar shape basic elements regularly distributed in space. The basic elements can refer to the fundamental micro-structures in generic natural images and are usually called textons (Song *et al*., 2004). In this paper, a 2D auto-correlation function is used to analyze the size and distribution of the basic texture elements. The auto-correlation function is defined as:

$$Y(\Delta x, \Delta y) = \frac{\sum_{ij} T(i,j)T(i+\Delta x, j+\Delta y)}{\sum_{ij} [T(i,j)]^2}$$

where i and j lie in a specific image T, this implies that elements outside T are zero, and position incremental d=(Δx, Δy) can be negative. For the given example texture image, the auto-correlation function reaches maximum 1 with d=0, and decreases with respect to positive or negative incremental position. The slope at the central peak can be thought as the texture grain. If the size of the basic element in textures is large, the decrease of the auto-correlation function value is relatively slow, but if the size of basic elements is small, the auto-correlation function decreases rapidly. If the texture is periodic or regular, then $Y(\Delta x,\Delta y)$ will reach maximum periodically. According to the distribution of peaks and valleys, the size and distribution of basic elements can be extracted.

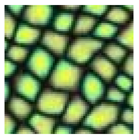
With regard to implementation, in order to reduce the search area determined by Δx and Δy and speed up computation, a crude threshold can be given empirically, then Δx and Δy can be varied within a given rang. If $i+\Delta x$ or $j+\Delta y$ is larger than the range, we employ the topological approach to reduce the width or height of textures so that search within a reasonable range can be guaranteed.

For Fig.1a, the search range for Δx and Δy respectively is 0~20. Experimental results for the auto-correlation function are given in Table 1 (in part).

Table 1 shows that the minimum of the auto-cor-

Table 1 Relationship between the search range for Δx and Δy and the auto-correlation function values

(Δx, Δy)	Auto-correlation value	(Δx, Δy)	Auto-correlation value
(0, 0)	1.00000	(10,11)	0.78597
(0, 3)	0.80682	(10,13)	0.76455
(1, 3)	0.78418	(11,13)	0.74678
(1, 5)	0.69043	(11,15)	0.73921
(2, 5)	0.67193	(12,15)	0.72929
(2, 7)	0.64254	(12,17)	0.74222
(3, 7)	0.63719	(13,17)	0.74295
(3, 9)	0.65351	(13,19)	0.72672
(4, 9)	0.66452	(14,19)	0.73164
(4, 11)	0.72928	(19,19)	0.83647


(a)

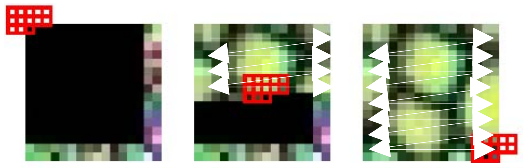

(b) (c) (d)

Fig.1 Texture synthesis procedure. (a) Example texture; (b) The first pixel synthesized; (c) The synthesized result in middle of synthesis; (d) Final texture synthesis result

relation, 0.63719, is reached at (Δx, Δy)=(3, 7). Among 399 auto-correlation values, it was discovered that there are three cases where values are larger than 0.9, 50 cases where values are larger than 0.8, 228 cases where values are larger than 0.7, 58 cases where values are larger than 0.6, and no cases at all where values are smaller than 0.6.

Computation results showed that the incremental position corresponding to the minimum of auto-correlation function is d=(3,7). The following formula was then used to calculate its geometrical average:

$$\overline{T}=\sqrt{[(\Delta x)^2+(\Delta y)^2]/2}\,.$$

The computation results in $\overline{T}$=5.38, and getting the closest integer, the size for the basic element is thus 5.

It was discovered in experiments that for the not well-structured natural textures, most of the auto-correlation function values are larger than 0.9. Thus, if the textures whose auto-correlation function values are larger than 0.9 occupy a large proportion (e.g., 80%), then it is certain that they are natural. In this case, a relatively small number can be used to reflect the size of the basic element. In general, when it is set to be 3, good results can be obtained. Also, computational efficiency can be improved if a larger size is used.

Improved Wei and Levoy (2000)'s texture synthesis method

Once the size of the basic element in textures has been determined, a comparison between neighbors can be made. In the Markov model, textures can be generated by a process based on the principle of local similar randomicity. Any pixel in textures can be regarded as a description of characteristics of neighboring points. If the size of basic elements obtained in the process of texture analysis is used as the size of the L shape, then the L_2 distance between different neighbouring areas can be calculated for matching. The L_2 distance between two identically shaped neighbouring areas N_1 and N_2 is defined as (Wei *et al.*, 2003):

$$d(N_1,N_2)=\sum \mathrm{sqrt}((f(p)-f(q))^2),\quad p\in N_1,\ q\in N_2,$$

where p and q are correspondents, $f(p)$ and $f(q)$ denote the feature values of the images respectively. Fig.1 describes the above procedure.

In Fig.1, L shape is the neighboring area of textures, whose size was obtained through texture analysis. Fig.1a is a example texture. Figs.1b~1d show the texture synthesis procedure with enlarged images. Fig.1b illustrates the first synthesized pixel. The neighbors are the noisy pixels created through topological relationships. As a result, the whole image is noisy; Fig.1c shows the result in the middle of synthesis. In this case, points in the neighbors are the texture pixels obtained in the previous synthesizing stages; Fig.1d shows the final synthesis result.

FRACTAL PATTERN DESIGN

Fractal patterns are self-similar patterns with infinite details in various scales, and reflect chaos and changeable characteristics in the natural world. Fractal based pattern design includes texture creation, colour design, and texture synthesis. Texture is the basis of texture synthesis. Its basic element is called unit texture. In fractal pattern, unit texture is a variable with fractal characteristics, which can be generated through certain fractal algorithms. Colour is the effect created by mixing, contrasting, and adjusting different hues, saturation and values in the HSV colour models. Pattern configuration means the layout and organization of fractal patterns. Fractal patterns can embody numerous, traditional beauty criteria, like balance, harmony, and symmetry, and also reflect some characteristics beyond these criteria.

Fractal number set

For the design of fractal patterns, Julia number set and Mandelbrot number set are most widely used. These two sets come from a non-linear mapping $x\rightarrow x^2+\mu$, and through transformation and iteration, the fractal computer images can be generated.

For the selection and implementation of fractal number sets, besides the basic Julia and Mandelbrot number sets, in order to make the obtained fractal patterns more valuable, we adopted the algorithm described in (Dewdney, 1986), mainly including Connett and Martin fractal number sets.

1. Connett number set

Connett number set is based on circles, but it can create numerous patterns based on squares. Connett number set completely abandons iteration, chooses to use the analytic formula of circles X^2+Y^2 to determine the colour at point (X, Y). Through scanning grid enclosed area, the fractal pattern for Connett number set can be obtained based on the odd or even characteristics of the rounded result calculated at each point using the circle formula.

Connett number set fractal patterns can be controlled by the following parameters: left bottom corner (*Corna*, *Cornb*) of the scanning area and the *Width* and *Height* of the scanning rectangles. The smaller the rectangular area, the closer the rectangle appears to the observer, and the greater the magnitude, the more is the pattern enlarged. This is a phenomenon similar to that of the pattern due to interferometry/diffraction in optics. However, the whole image appears dull. On the contrary, the larger the rectangular area, the better the overall characteristics of the fractal patterns reflected. Different *Corna* and *Cornb* can lead to different increments and thus, different fractal patterns.

Fig.2 shows three fractal patterns created by Connett number set.

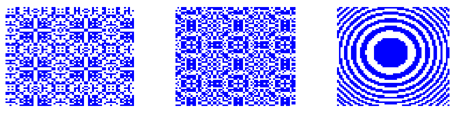


Fig.2 Three fractal patterns created by Connett number set with different parameters. (a) *Corna*=50, *Cornb*=35, *Width*=840, *Height*=840; (b) *Corna*=555, *Cornb*=555, *Width* =80, *Height*=80; (c) *Corna*=555, *Cornb*=555, *Width*=10, *Height*=10

In Figs.2a~2c, the rectangle size becomes smaller and smaller. Figs.2a and 2b show that different values of *Corna* and *Cornb* can respectively yield different fractal patterns. Comparing Fig.2b with Fig.2c, it shows that with the same values for *Corna* and *Cornb*, decreasing values for *Width* and *Height* has an enlarging effect.

2. Martin number set

Martin number set was inspired by Mandelbrot number set and applies the idea of iteration of Mandelbrot number set, but its iteration is different from that of Mandelbrot. Martin number set starts its iteration based on a single initial real number, while Mandelbrot number set starts its iteration based on a set of complex numbers spread on a plane.

The following are two pairs of formula for Martin number set:

$$X=Y-\mathrm{sign}(X)\times\mathrm{abs}(B\times X-C)/2,\quad Y=A-X, \tag{1}$$
$$X=Y-\sin(X),\quad Y=A-X. \tag{2}$$

In Eq.(1), the sign function sign(X) can take 1 or −1, depending on whether X is positive or negative. The function abs($B\times X-C$) takes the absolute value of $B\times X-C$. Changing parameters A, B, and C can create different fractal patterns. The parameter A controls the magnitude of shrinkage and enlargement of the patterns, the larger the parameter A, the smaller the shrinking and enlarging magnitude. The parameter B controls the appearance of the patterns; and C controls the size of the pattern (proliferation extent), the larger the parameter C, the better the pattern appears in a larger area. Eq.(2) just applies a parameter A and a simple sine function to generate fractal patterns. Our experience showed that when the difference between A and π is smaller than 0.07, Martin number set can create ideal fractal patterns.

In the two formulas, X and Y can be initialized as 0. Meantime, the iteration number also must be initialised and determines the number of pixels in fractal patterns. The larger the iteration number is set, the more fine-grained the fractal patterns are. Fig.3 illustrates four fractal patterns created by Martin number set.

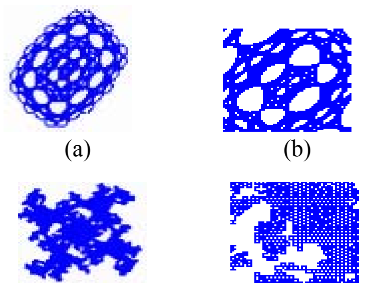


Fig.3 Fractal patterns created by Martin number set. (a) and (b) The patterns created using Eq.(1); (c) and (d) The patterns created using Eq.(2)

Figs.3a and 3b are the created fractal patterns using Eq.(1). The former is the original, and the latter is an enlargement of a certain area, where the relevant parameters are set as: A=4, B=−185, C=−269, iteration number=100000. Figs.3c and 3d are the generated fractal patterns using Eq.(2). The former is the original, and the latter is an enlargement of a certain area where the following parameter values were used: A=3.2, iteration number=200000.

Synthesis fractal procedure

The idea of synthesis fractals is to overcome the shortcomings of colors and the dull textures in the process of traditional fractal pattern creation through various rendering rules, so as to apply colorful textures into fractal patterns. In the process of synthesis fractals, the geometric patterns that possess fractal structure can be generated first. Then sample texture can be rendered into the fractal structures through texture synthesis, so that the rendered patterns possess both the geometric structure of fractals and the characteristics of the sample texture.

From the above introduction we know that fractal patterns are gradually implemented as a series of points through iteration. If the colours need to be considered for rendering in the patterns, especially in order to embody some realistic colour perception, it is not only necessary to consider the geometry, but the harmony of colours as well. This process to complete texture synthesizing with alterable colour is called render mapping.

In the process of traditional fractal pattern generation, it is necessary to set up the rendering rules based on the colours of the convergence point. Although the rendering rules are changeable, the patterns obtained still lack certain regularity. In addition, for fractal patterns with special design requirement, even if they have the geometric structure of fractal patterns, the colour and texture of the patterns may not satisfy the design requirement. Therefore, we introduce texture synthesis with colour changeable mechanism for the rendering rules for colours and textures.

The process of rendering is according to the information of point sets obtained by the iterative procedure. For each point in the point set, a mask is first set. The improved Wei and Levoy (2000)'s algorithm is then used for synthesizing textures and, at the same time, HSL color space is used for controlling the color rendering. Two different texture rendering rules are:

Rule 1 (rendering based on Mask bits) The rendering process focuses on the points with the same bits with the Mask, neglecting all others. This rule is equivalent to the rendering of the geometric outline of fractal patterns. Although the rendering based on the rule is fast, the final rendering result is still limited, mainly because the neighbouring information used for the neighbouring similarity matching is incomplete (including points not yet synthesized).

Rule 2 (rendering based on the boundary of fractal patterns) The rendering process focuses on the points within the external rectangle of the fractal pattern. The final synthesized result only considers the points inside the boundary and having the same bits as Mask. The texture information for other points is only needed for temporary storage so that the synthetic coherence is maintained. This is equivalent to rendering the whole bounded rectangle, and is relatively slow, but the final rendering result is better than that using Rule 1.

In order to generate richer texture rendering result, we transform the colour of each pixel from RGB model to HSL model which emphasizes the relationship between light intensity and color variation and depicts hue (H), saturation (S), and light intensity (L) respectively. Appropriate configuration of H can lead to various kinds of fractal patterns with similar texture structure as texture rendering but different colour. For texture with strong regularity, it is necessary to select an appropriate neighboring size. If the selected size is exactly the same as that of the basic texture element, the rendering result is best; for stochastic texture, the selection of neighbouring area will not influence too much the final rendering result.

In summary, assume that sample texture is T_i, synthesis texture is T_o, two rendering rules are Rule 1 and Rule 2 respectively, the whole fractal process synthesis is described as follows:

```
Function SynthesisFractal($T_i$, $T_o$, $Offset$, $Rule$)
{
   $T_o$ ← CreateFractalStructure()
   $S$ ← CalculateParticleSize($T_i$)
   Switch($Rule$)
         Case Rule 1:
               MaskSythesis($T_i$, $T_o$, $S$)
         Case Rule 2:
               NormalSysthesis($T_i$, $T_o$, $S$)
    ChangeHUE($T_o$, $Offset$)
}
```

```
Function MaskSythesis($T_i$, $T_o$, $S$)
{
  For Each Pixel($X$, $Y$) In $T_o$
      If Mask($X$, $Y$)=TRUE Then
          $P$←CreateNeighborHood($T_o$, $X$, $Y$, $S$)
          Pixel($X$, $Y$)←FindBestMatch($P$, $T_i$, $S$)
}
Function FindBestMatch($P$, $T_i$, $S$)
{
  MatchValue←MaxMatchValue
  TMathValue← MaxMatchValue
  BestMatch←NULL
  For Each Pixel($X$, $Y$) In $T_i$
      $P'$←CreateNeighborHood($T_i$, $X$, $Y$, $S$)
      TMatchValue←CalculateL2Distance($P$, $P'$)
      If MatchValue>TmatchValue
          MatchValue←TMatchValue
          BestMatch←Pixel($X$, $Y$)
      Return BestMatch;
}
```

Function CreateFractalStructure() generates the original patterns with fractal structures; Calculate-ParticelSize() calculates the size of the basic element in the sample texture; ChangeHUE() changes the hues of the synthesized texture, enriching the rendering effect; MaskSythesis() and Normal-Systhesis() render the textures according to different rendering rules, whose implementation can refer to the formalisation described above; Function FindBestMatch() implements neighbouring area search and matching; CalculateL2Distance() calculates L_2 distance between neighboring areas.

EXPERIMENTAL RESULTS

Example 1

For Martin number set, Eq.(1) is used as fractal formula where the following parameter values are used: A=−398, B=5, C=−452, the number of iterations=5000, incremental for H=0, neighbouring size=3. Fig.4b shows the result for the application of rendering Rule 1, which took 35 s on a PC; Fig.4c illustrates the result for the application of rendering Rule 2, which takes an evolutionary time of 27 s. Sample pattern Fig.4a is of 32×32 pixels. The size of fractal patterns is of 100×100 pixels.

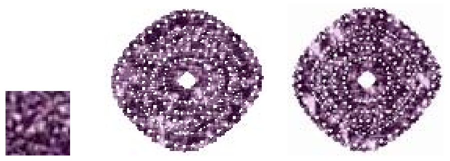


Fig.4 Rendering results with Martin number set. (a) Example texture; (b) The result based on Rule 1; (c) The result based on Rule 2

Example 2

Connett number set was applied as the fractal formula where the following parameter values were used: *Corna*=555, *Cornb*=555, *Width*=80, *Height*=80, neighbouring size=5. Rule 2 was employed for rendering. Fig.5b illustrates the result with no incremental value; Fig.5c illustrates the result with the incremental value equal to 50; Fig.5d illustrates the result with incremental value equal to 100. Fig.5a is a sample with a size of 32×32 pixels. The size of fractal patterns is of 100×100 pixels, which took 41 s.

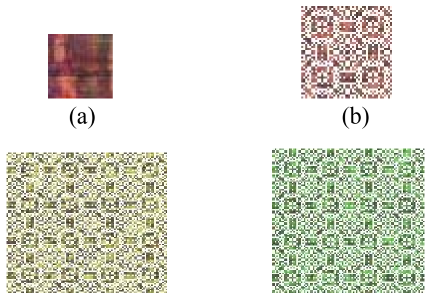


Fig.5 Rendering results with Connett number set. (a) Example texture; (b) ∇h=0; (c) ∇h=50; (d) ∇h=100

Example 3

Fig.6 illustrates the fractal patterns created by various Martin and Connett number sets with different parameters and different rendering rules.

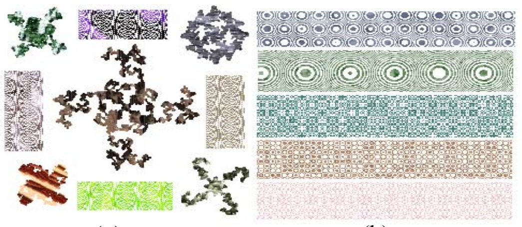


Fig.6 Rending results with different fractal number set and parameters. (a) Different fractal number sets; (b) Different parameters and hue values

CONCLUSION

This paper has improved Wei and Levoy (2000)'s point based matching algorithm for more accurate and efficient texture synthesis and design. The improvement lies in the application of 2D auto-correlation function for the size analysis of the basic elements in textures. The resulting algorithm has the characteristics of automatic recognition of basic pattern element and determines the size of the L neighboring area. Then we presented fractal pattern design approach based on pattern rendering, synthetically applying the techniques of texture synthesis and fractal pattern design. Fractal pattern with arbitrary sample pattern characteristics can be rapidly obtained through pattern exaggeration. This technique provides a novel train of thought to pattern design, and also finds new application area for the research of texture synthesis in the fields of the art, spinning and weaving, and dying industry which have to deal with pattern design.

Further research will be to design better pattern rendering algorithm, automatically create fractal patterns with specified characteristics and apply these techniques to actual applications.

References

Barni, M., Bartolini, F., Cappellini, V., 2000. Image processing for virtual restoration of artworks. *IEEE Multimedia*, **7**(2):34-37. [doi:10.1109/93.848424]

Bertalmio, M., Vese, L., Sapiro, G., Osher, S., 2003. Simultaneous Structure and Texture Image Inpainting. Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, p.1063-1069.

Cohen, M.F., Shade, J., Hiller, S., Deussen, O., 2003. Wang tiles for image and texture generation. *ACM Transactions on Graphics*, **22**(3):287-294. [doi:10.1145/882262.882265]

Daniilia, S., Sotiropoulou, S., Bikiaris, D., 2000. Panselinos' Byzantine wall paintings in the Protaton Church, Mount Athos, Greece: a technical examination. *Journal of Cultural Heritage*, **1**(2):91-110. [doi:10.1016/S1296-2074(00)00164-3]

Dewdney, A.K., 1986. Wallpaper for the mind: computer images that are almost, but not quite, repetitive. *Scientific American*, **255**(Sept.):14-23.

Efros, A.A., Leung, T.K., 1999. Texture Synthesis by Non-parametric Sampling. International Conference on Computer Vision (Sept. 1999), p.1033-1038.

Efros, A.A., Freeman, W.T., 2001. Image Quilting for Texture Synthesis. Proceedings of Siggraph'2001, p.341-346.

Kwatra, V., Essa, I., Schödl, A., Turk, G., Bobick, A., 2003. Graphcut Textures: Image and Video Synthesis using Graph Cuts. ACM Transactions on Graphics, Siggraph'2003, p.277-286.

Song, P., Meng, X.X., Tu, C.H., Yang, C.L., 2004. Texton-based texture synthesis. *Proceedings of SPIE*, **5444**:138-144. [doi:10.1117/12.561123]

Wang, X., 2003. Algorithmic construction technique for the computer aided design of fractal patterns. *Journal of South China Normal University (Natural Science)*, **1**:46-51.

Wei, L.Y., Levoy, M., 2000. Fast Texture Synthesis Using Tree-structured Vector Quantization. Proceedings of Siggraph'2000, p.479-488.

Wei, B.G., Liu, Y.H., Pan, Y.H., 2003. Using hybrid knowledge engineering and image processing in color virtual restoration of ancient murals. *IEEE Transactions on Knowledge and Data Engineering*, **15**(5):1338-1343. [doi:10.1109/TKDE.2003.1232282]