# Multi-Layer Inpainting on Chinese Artwork

Timothy K. Shih., Rong-Chi Chang, Liang-Chen Lu and Huan-Chi Huang
Multimedia Information Network Lab
Department of Computer Science and Information Engineering
Tamkang University, Taiwan, ROC
E-mail: tshih@cs.tku.edu.tw

## ABSTRACT

Digital inpainting uses spatial or frequency information to restore partially damaged/removed photos and artworks. Most restoration algorithms take a picture as a single layer. We propose a new approach, which subdivides a Chinese painting into several layers. Each layer is inpainted separately. The layer fusion mechanism then finds the optimal inpaint among layers, which are restored layer-by-layer. We apply the algorithm on Chinese and western drawing. The result shows a high PSNR value as well as a high user satisfaction. The demonstration of our work is available at: *http://www.mine.tku.edu.tw/demos/inpaint.*
**Key words:** digital inpainting, image restoration, multi-layer inpainting, image processing

## 1. INTRODUCTION

Automatic digital inpainting uses spatial information for color interpolation as well as applies intelligent mechanisms on frequency domain, especially for large damaged blocks. Current techniques base on the extrapolation of neighboring pixels, recovery of edges, curvature-driven diffusions (according to the connectivity principle in vision psychology) [3], texture synthesis [9], and inpainting from multiple view points (i.e., image from movie, or image from different time and view point). A fast inpainting algorithm was proposed in [4]. Efficiency of the proposed method [4] is two to three orders of magnitude faster than those using partial differential equations. The mechanism presented in [1] automatically inpaints a user-selected region, by using surrounding information. Another paper [2] takes the ideas from classical fluid dynamics to propagate isophote lines. Inpainting can be used in attacking a visible watermark [5]. The system discussed in [5] allows users to select a watermark area, and to produce an approximation to the original picture. The technique discussed in [6] combines texture and diffusion, by using DCT to compute H (high frequency) and L (low frequency) of the entire image. Diffusion is used to handle L. Multi-level texture synthesis is used to handle H. The results are summed up. Inpainting on large blocks are presented in [7, 8, 9]. The exemplar-based synthesis approach [7] used both textural and structural information and recovers user selected area which covers 19% of the image. The mechanism presented in [8] also takes into consideration the textural and structural information. A low computation cost mechanism based on texture synthesis with spatial-temporal consideration is found in [9].

Most inpainting algorithms take the damaged picture as a single layer, and computes inpaint data based on the single layer. The approach may use irrelevant information on restoring an object, without considering the separation of the object and its background. In Chinese painting, an artist usually draws the background using a light color. The middle layer is in a darker color. And the front layer is in an even darker color. Each layer contains different objects (e.g., far mountain, near mountain, trees, and people). Western classical drawing may have the same approach. It makes sense to separate a drawing into different layers. And, inpainting should be proceeded separately. Finally, a fusion algorithm should be used to combine the results. The algorithms proposed in this paper is implemented and tested on more than 2000 pictures, including Chinese and western painting, photos, and cartoon drawings. Interested readers can visit the following Web site for a demonstration: http://www.mine.tku.edu.tw/demos/inpaint.

## 2. LAYER SEPARATION

Color region separation is a difficult challenge, if the objective is to precisely detect the boundary of objects. In digital inpainting, it is impossible to restore an image to one hundred percents since information is lost. An approximation approach to separate objects in to different layers is reasonable for inpainting. We use a layer separation algorithm to divide a painting into several layers. It is difficult to decide the threshold of separation. However, pixels of similar colors can be divided into groups, which represent layers.

***Definition:*** A *dominant color* is a pixel color which occurs in a picture for more than a percentage threshold, $\rho$. A *dominant mean color* is the average of a group of pixel colors, which contain at least one dominant color and each color in the group differs by at most $\delta_1$.

The percentage threshold $\rho$ and the color difference $\delta_1$ can be defined according to the characteristics of a picture. A typical situation is to set $\rho = 5$ and $\delta_1 = 20$. That is, to find a dominant mean color, we define a dominant color which

0-7803-8603-5/04/$20.00 ©2004 IEEE

have pixels occurs in a picture for at least 5 percents. And, other pixels differ to the dominant color by at most 20. The use of $\delta_1$ depends on the color space used. For the RGB color space, values are ranged from 0 to 255. The difference can be calculated from the average of R, G, and B (not recommended). On the other hand, if YUV or HIS is used, the difference can be calculated from a linear combination of values. It is difficult to set the thresholds of how dark of pixels at each level. But, according to Chinese painting, lightest areas are background. Darker areas are foreground or details. The following algorithm results in **K** pictures:

***Algorithm*: Layer Separation**
Let **DIB** be a damaged image block
Let **DIBLayer[K]** be a set of damaged image layers
Let $\rho$ be a threshold of percentage
Let $\delta_1$ be a threshold of color difference
**Layer_Separation** (block **DIB,** integer **K**) {
1.Compute the color histogram of **DIB**
2.Sort the colors, select the top **K** dominant mean colors (depends on $\rho$ and $\delta_1$)
3.Use the K-mean classification algorithm, and set the seeds to the top **K** dominant mean colors, and classify pixels into **K** groups
4.Separate **DIB** into **K** layers in **DIBLayer[K]**. Compute the mean color of non-damaged pixels in each layer.
}

The results are stored in **DIBLayer[K]** , which is used by an inpainting algorithm discussed in the next section. Note that, we use a YUV color space, with a focus on Y. The threshold $\delta_1$ can be adjusted. Mostly, it is set to 20 to obtain a reasonable result. The threshold $\rho$ is set to 5 by default. This value is also adjustable. The algorithm uses K-mean for classification (for the sack of simplicity), which results in a reasonable good separation (see figure 1 and our website).

## 3. THE INPAINTING ALGORITHM

We realize that different portion of a picture contains different levels of details. We use a multi-resolution strategy, which looks at the details, and decide what surrounding information to use. Level of details can be indicated by the variance of color distribution in a portion of image. We use another percentage threshold $\alpha$ for color variance. Distance of color is treated the same as that used for $\delta_1$. Value of color variance ranges from zero to several thousands depends on pictures. According to our experiment, $\alpha = 80\%$ results in a good result. Two additional percentage thresholds, $\beta_1$ and $\beta_2$, are used in the inpainting algorithm for different situations of decomposition. An input damaged picture **DIBk** is divided into several image blocks (i.e., **IB**s). If the percentage of damaged pixels in an **IB** is too high, using surrounding color information to fix a pixel is less realistic as compared to using a global average color. In some severe cases, it is impossible to use neighborhood colors. Note that, both thresholds are adjustable for the sake of analysis. The recursive algorithm iterates through each of the **IB**s in a damaged picture. If the color variance of IB is below the percentage threshold $\alpha$, there is not much difference of pixels in the **IB**. No subdivision is required (i.e., no need of looking at the next level of details). Thus, the algorithm further divides **IB** into several pixel blocks (i.e., **PB**s). If the percentage of damaged pixels in a **PB** is too high (i.e., greater than $\beta_2$), the mean color of **IB** is used. One example is that the entire **PB** is damaged (thus we need to use the mean color of **IB**). Alternatively, if the percentage is still high (i.e., greater than $\beta_1$), the mean color of **PB** is used. Note that, the computation of mean colors does not take damaged pixels into the account. If the percentage is low, neighbor pixels are used for inpainting. Finally, if the color variance of **IB** is not below the threshold $\alpha$, the algorithm is called recursively to handle the next level of details. The following algorithm shows the details:

***Algorithm*: Multi Resolution Inpainting**
Let **DIBk** be a damaged image block
Let $\alpha$ be a threshold of variance
Let $\beta_1$, $\beta_2$ be a threshold of percentage, $\beta_1 < \beta_2$
**Multi_Resolution_Inpainting**(block **DIBk**) {
  divide **DIBk** into $n \times n$ image blocks
  For each image block **IB** {
   let *var* be the color variance of **IB**
   If $var < \alpha$ then {
    Let **PB** be an $x \times y$ pixel block in **IB**
    For each **PB** in the image block {
      If the percentage of damaged pixels in **PB** > $\beta_2$
          inpaint style = Mean of **IB**
      else if percentage of damaged pixels in **PB** > $\beta_1$
          inpaint style = Mean of **PB**
      else
          inpaint style = Neighboring
      Inpaint PB use its inpaint style
      }
    }
   else call **Multi_Resolution_Inpainting(IB)**
  }
}

After a damaged picture is decomposed into **K** layers, we use the above Multi Resolution Inpainting algorithm to inpaint damaged areas in each layer of the picture (i.e., **DIBLayer[K]**). According to the decomposition, the first layer has the lightest colors, which represent a far background. The darker the color the higher chance of a fore background. However, to combine the separated layers after inpainting, we need a fusion algorithm. The fusion algorithm follows a strategy. For each damaged pixel to be inpainted, two consecutive layers are compared. A window with pixels in a distance **D** with respect to an inpainted pixel **P** is used. The function $\mu$**[P]** computes percentages of useful pixels within distance **D** is applied to each inpainted pixel. Depending on the percentages, a layer is selected. Useful

pixels are non-inpainted pixels from the original image, with respect to a layer. The far ground is firstly placed in a blank paper. The picture is restored with a darker layer step-by-step. In Chinese painting, light far grounds are drawn first, followed by darker grounds. The restoring strategy has a similar effect.

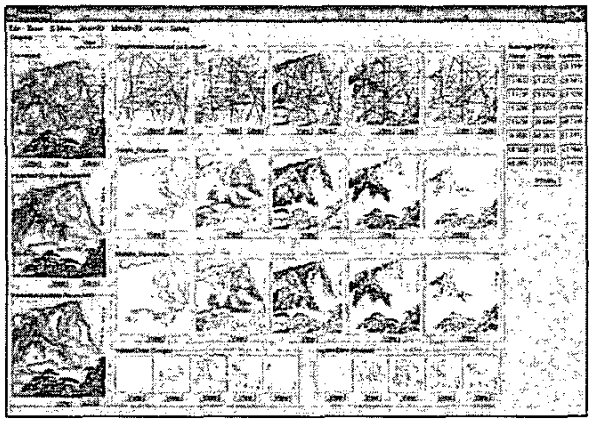

**Figure 1: A Multi-Layer Inpainting Tool**
http://www.mine.tku.edu.tw/demos/inpaint

***Algorithm*: Multi Layer Fusion**
Let **DIBLayer[K]** be a set of damaged image layers
Let **PIC** be a picture buffer
Let **D** be a distance representing a window size
Let $\mu$**[P]** be a percentage of useful pixels within a distance **D**, with respect to a pixel **P**

```
Multi_Layer_Fusion(block layers DIBLayer[K]) {
Copy DIBLayer[1] to PIC
For layer = 1 to K-1 {
  For each damaged pixel P {
    If μ[P] of layer > μ[P] of layer + 1
      Inpaint P in PIC using the pixel in layer
    else
      Inpaint P in PIC using the pixel in layer + 1
    }
  }
}
```

Thus, the inpainting algorithm has multiple layers and multiple resolutions. On the other hand, we also implemented a simple inpainting strategy, which does not decompose the damaged image according to its details. This single resolution approach may have a side effect that inpainting errors are propagated. In general, the multiple resolution approach has better PSNR values compared to the single resolution approach. We discuss the difference next.

## 4. ANALYSIS

Some test results of multi-layer multi-resolution inpainting are shown in figure 2 to figure 4. Different categories of Chinese paintings are tested on different types of damages. To test the inpainting results, we decompose a picture using a complete quad-tree representation. Each level of decomposition equally divides an area into 4 quadrants. The number of levels depends on the actual size of a picture. As a practical situation, we decompose pictures up to 8 levels for evaluation. The values of PSNR become high as the level number increases. This is due to the fact that, the percentages of noise in a picture is relatively low. In some tests, if the entire picture is covered by noise, the PSNR values increase slowly when the level number increases. In addition to Chinese paintings, we also use photos and cartoon drawings. We tested 1500 pictures with the PSNR values of multi-layer multi-resolution inpainted pictures shown in table 1. In addition, a single resolution approach is used. The average PSNR values of an entire picture (i.e., level 1) are higher than 30 dB. Multi-resolution inpainting out performs the single resolution approach.

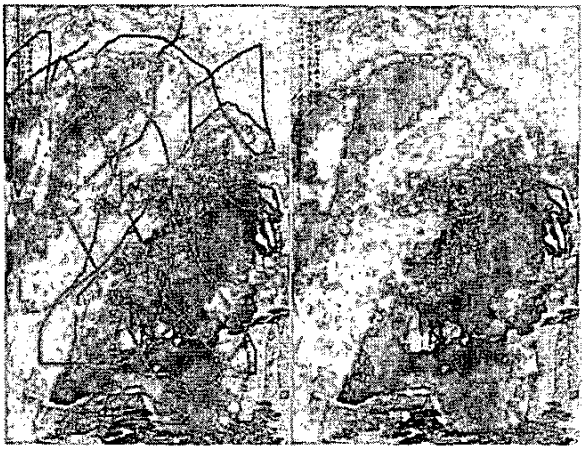

**Figure 2: Damaged and Inpainted Pictures (Landscape)**

**Table 1: Average PSNR values (dB) with 1500 pictures**

| Level | Damaged | Single | Multiple |
|---|---|---|---|
| 1 | 17.52487 | 30.28027 | 33.5023 |
| 2 | 17.75953 | 31.01713 | 34.2935 |
| 3 | 19.3279 | 35.18993 | 38.2828 |
| 4 | 28.81017 | 45.48623 | 47.98687 |
| 5 | 46.56983 | 60.05327 | 61.7763 |
| 6 | 64.14183 | 73.6296 | 74.7055 |
| 7 | 77.57397 | 83.7065 | 84.32603 |
| 8 | 86.3982 | 90.18937 | 90.53943 |

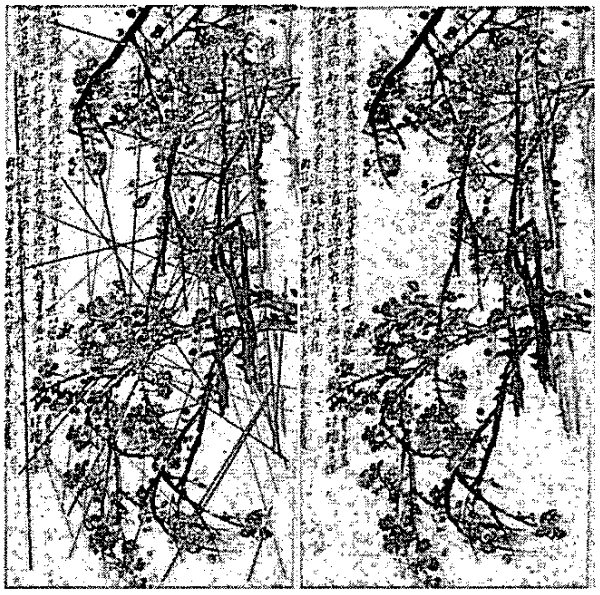

**Figure 3: Damaged and Inpainted Pictures (Flowers)**

Among the three categories of pictures, inpainted photos show the best result. The average PSNR at level 1 is 38.6687 for multi-resolution and 34.7254 for single resolution. Chinese and western paintings have average PSNRs equal to 34.0350 (multiple) and 30.8633 (single). The averages of cartoon drawing is 27.8032 (multiple) and 25.2521 (single). Each category contains 500 pictures. Noises are randomly generated (similar to those shown in figure 3 and 4). Cartoon drawings have crisp boundaries. Without using a line detection algorithm, it is hard to predict the continuity of color distribution.

**Figure 4: Damaged and Inpainted Pictures (People)**

In our experiment, each picture is decomposed into 5 layers (i.e., K = 5). It is very difficult to choose thresholds. We tested different combinations. The result shown in table 1 uses the following thresholds:

$\rho = 5\%$ and $\delta_1 = 20$
$\alpha = 80\%$, $\beta_1 = 90\%$, and $\beta_2 = 95\%$

The above thresholds of $\alpha$, $\beta_1$, and $\beta_2$ are the optimal solution of

$\alpha = 50\%, 60\%, 70\%, 80\%, 90\%$
$\beta_1 = 60\%, 65\%, 70\%, 75\%, 80\%, 85\%, 90\%$
$\beta_2 = 95\%$

The selection of $\beta_2$ is to test the mean color of the outside image block. Unless a pixel block is seriously damaged, otherwise, the mean color should be used. Thus, the selection of $\beta_2$ should be high. Since $\beta_1 < \beta_2$, we select the values of $\beta_1$ accordingly. The threshold $\alpha$ is to check the variance. We try to cover a wide spectrum of variances.

## 5. CONCLUSIONS

We proposed a new approach, which inpaints a Chinese artwork according to how it was drawn. A painting is decomposed into several layers and inpainted. The inpainting result is obtained by a layer fusion strategy. The proposed inpainting algorithm work very well on restoring both Chinese and western damaged artworks. The proposed algorithm is implemented on an inpainting tool, which will be released soon to the public.

## REFERENCES


[1] M. Bertalmio, G. Sapiro, V. Caselles and C. Ballester, "Image Inpainting," in Proceedings of the ACM SIGGRAPH Conference on Computer Graphics, 2000, SIGGRAPH 2000, 2000, pp.417-424.

[2] M. Bertalmio, A. Bertozzi, G. Sapiro, "Navier-Stokes, Fluid-Dynamics and Image and Video Inpainting," in Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001, pp. I355-I362.

[3] T. F. Chan and J. Shen, "Nontexture Inpainting by Curvature-Driven Diffusions," Journal of Visual Communication and Image Representation, V. 12, N. 4, 2001, pp. 436-449.

[4] Manuel M. Oliveira, Brian Bowen, Richard McKenna, Yu-Sung Chang, "Fast Digital Image Inpainting," in Proceedings of the International Conference on Visualization, Imaging and Image Processing (VIIP 2001), 2001, pp. 261-266.

[5] C.-H. Huang; J.-L. Wu, "Inpainting attacks against visible watermarking schemes," in Proceedings of Spie, the International Society for Optical Engineering, 2001, no. 4314, pp. 376-384.

[6] Yamauchi, H.; Haber, J.; Seidel, H.-P.; "Image restoration using multiresolution texture synthesis and image inpainting", Computer Graphics International, 2003, pp. 108-113.

[7] Criminisi, A.; Perez, P.; Toyama, K.; "Object removal by exemplar-based inpainting", IEEE Computer Vision and Pattern Recognition, 2003. vol. 2, 2003, pp. 721 -728.

[8] Bertalmio, M.; Vese, L.; Sapiro, G.; Osher, S.; "Simultaneous structure and texture image inpainting", IEEE Transactions on Image Processing, vol. 12, no. 8 , 2003 pp. 882 -889.

[9] Raphael Bornard, Emmanuelle Lecan, Louis Laborelli, Jean-Hugues Chenot, "Missing Data Correction in Still Images and Image Sequences," ACM Multimedia'02, Juan-les-Pins, France, December 1-6, 2002.