Exemplar-based Image Inpainting base on Structure Construction

Jason C. Hung[1]
Department of Information Technology, The Overseas Chinese Institute of Technology, Taichung, Taiwan
Email: jhung@ocit.edu.tw

Chun-Hong Hwang[2], Yi-Chun Liao[3], Nick C. Tang[4], Ta-Jen Chen [4]
[2]Department of Computer Information and Network Engeering, Lunghwa University of Science and Technology, Taoyuan County, Taiwan
[3]Department of Computer Science and Information Engineering, China University of Technology, Hsinchu County, Taiwan
[4]Department of Computer Science and Information Engineering, Tamkang University, Taipei County, Taiwan
Email: ech_liao@cute.edu.tw[3], ch.huang@mail.lhu.edu.tw[2], simon129@gmail.com

***Abstract*—Image inpainting technique is that repairs damaged area or remove areas in an image. In order to deal with this kind of problems, not only a robust image inpainting algorithm should be used, technique of structure generation is also needs to adopt in inpainting procedure. In this paper, we extend an exemplar-based image inpainting method by incorporating Bézier curves to construct missing edge information. Before we apply our inpainting procedure to a damaged image, mean shift segmentation is used to extract contours of each region. After the structure information of an image is determined, destroyed contours will be connected in curve fitting process and damaged regions will be inpainted by using exemplar-based inpainting method. The experimental results show that our proposed method could deal with several kinds of pictures well.**

***Index Terms*—Image Inpainting, Color segmentation, Bézier curves, Mean Shift Segmentation**

I. INTRODUCTION

Ages ago, people are already preserving their visual works carefully. Now, because the popularity of computers, people often keep their arts, pictures, even old films or cartoons in their computers. By the great power of modern computers, we can inpaint digitalized images.

Image inpainting/image completion [9,10,11] is a technique to restore/complete the area of a removed object which is manually selected by the users. The technique produces a reasonably good quality of output on still images. Although there are earlier approaches which focus on removing only small well selected areas on photos, the work reported in [9,10] produces fairly good results in general cases, especially when applied to large continuous areas.

Inpainting is also known as retouching is the activity consists of filling in the missing areas or modifying the damaged ones in a non-detectable way for an observer not familiar with the original images [1]. Shih-Chang Hsia [2] use a noise detective filter to determine whether a pixel is a noisy pixel or not. If it is a noisy pixel, replace it by the average of non-noisy pixels. If it is a non-noisy pixel, simply skip the process and preserve that pixel.

Bertalmio [1] fill in the missing date by the property of isophotes (lines of equal gray value). Inspired by the work of Bertalmio, T. Chan and J. Shen proposed the Total Variational (TV) inpainting model [3], then extend it to Curvature-Driven Diffusion model [4] by defining the strength of diffusion process. Above methods needed to solve Partial Differential Equations(PDE). It is a very complex and time- consuming process that takes minutes.

Manuel M. Oliveira [5] uses a diffusion kernel to convolve the inpainting domain try to reduce the complex of PDE. However this method doesn't preserve the isophotes directions very well, some high-gradient image area must be manually select before inpainting to prevent blur.

But, Among numerous inpainting techniques, exemplar-based inpainting [9] is most often used tp complete damaged area or remove objects in an image. The algorithm proposed in [9] deal with the topology of inpainting by using a priority value for each patch. The priority value P is evaluated by function P = C * D. The confidence value C value is computed according to the percentage of "real" information which is used in the filling process. The data term D is computed by the inner product of an orthogonal-gradient and a unit vector orthogonal to the front damaged area.

As shown in figure 1, the work of our inpainting procedure starts with robust algorithm of color segmentation: mean shift segmentation. After we separate out structure information form image, the missing contour could be repaired by using Bézier curve. Finally, the exemplar-based image inpaintng method would be applied to recover all information of damaged area from other source area.

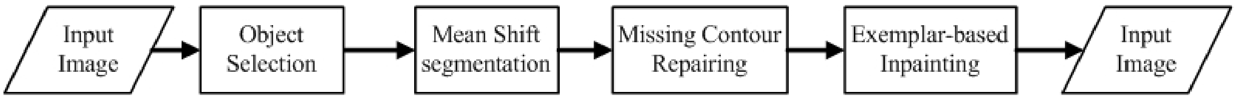


Figure 1. System flow chart

The basic condition under the both side of the inpainting domain has the same amount of *n* contour lines as figure 2 shows. And then to extend the work for salving the situation of (*n+1)* contour lines to *(n)* contour lines which in both side respectively as figure 6 shows. In this paper, we first construct the contour lines of image requiring inpainting, and then estimate the slope of contour line. Use them to estimate how they should extend and where two extended contour lines will meet.

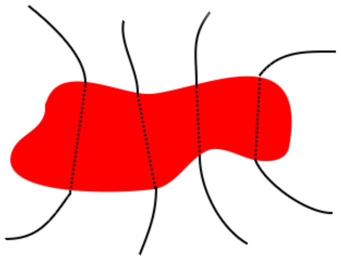


Figure 2. The basic idea to recover the contour lines.

This paper is organized as the following. We discussed the essential techniques in section 2. The detailed description of our proposed algorithm will be described in section 3, with results and conclusion in sections 4 and 5, respectively.

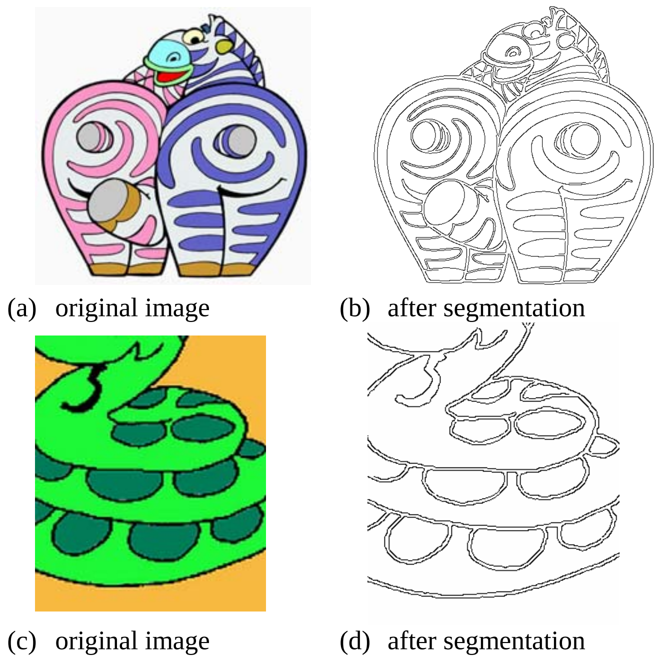


Figure 3. The template image

II. Essential techniques

In order to provide a flawless result of inpainting, there are several kinds of techniques such as mean shift segmentation, exemplar-based image inpainting and Bézier curves should be adopted.

A. Color Segmentation by Mean-Shift Segmentation

Before we start, we have to construct the contour lines of the image required for inpainting. In the field of low level computer vision, many color segmentation and edge detection method had been proposed. From the well-known Sobel edge detection to Canny edge detection, most of them are too sensitive to texture of cartoons. That's why we choose Mean-Shifts based color segmentation [6, 7] to minimize the colors and textures and then we can produce the contour lines using edge detection.

B. Exemplar-based Image Inpainting Revised

The contribution from [9] introduced two important fundamental techniques of image inpainting: priority map (i.e., P(p)) and conference term (i.e., C(p)). The priority map, P(p) = C(p)*D(p), uses a data term (i.e., D(p)) in additional to using the confidence term. The data term is based on the product of isophote and a unit vector orthogonal to the front of an inpainted area. The confidence term favors out-pointing regions (e.g., sharp bands) in the inpainted area. Thus, smooth or near circular inpainting area results. On the other hand, the data term gives a high priority to inpainted area which has a potential to copy structural information from the source area. The data term and the confidence term together results in a balance to select the best patch to be inpainted. When the inpainting algorithm selects a patch from the source area, sum of squared differences (SSD) using the CIE Lab color space is implemented. In addition to using the similar approach, we introduce a new matching strategy, based on Edge Crispening and a new concept of *patch template*, to make sure that the best structural similarity is achieved.

Our discussion strictly follows the notations introduced in [1]. Readers should refer to this important article. Let $\boldsymbol{I}$ be the original image (or a frame in a video) which includes a target area, denoted by Ω, to be inpainted and a source area, denoted by Φ, where patches are searched and used. Hence, $\boldsymbol{I} = \Phi \cup \Omega$. A simple region segmentation algorithm based on the CIE Lab color space is used to convert $\boldsymbol{I}$ to $\boldsymbol{I'}$. Let p_i and p_j be pixels and s_i and s_j be segments.

$\forall p_i \in I, \forall p_j \in I, p_i \neq p_j$ and p_i is adjacent to p_j,

$SSD_{\text{CIE Lab}}(p_i, p_j) < \delta_c \Rightarrow$ make p_i , p_j in the same segment

$\forall s_i \in I, \forall s_j \in I, s_i$ is adjacent to s_j,

$pn(s_i) - pn(s_j) < \delta_{pn} \Rightarrow$ make s_i , s_j in the same segment where *pn(s)* computes the number of pixels in a segment. And, *SSD* $_{CIE\ Lab}$ (p_i , p_j) calculates the sum of squared differences using the CIE Lab color space. According to experiments, If δ_c is less than 3, the segmentation is hardly perceptible. If it is greater than 6, the result is not useful. We set the threshold δ_c to 3 simply to keep more segments for edge analysis. The second threshold δ_{pn} is different according to video. We manually adjust this threshold for each video. Thus, ***I'*** is the result of color segmentation by using mean shift segmentation. The Edge Crispening algorithm (not used in [9]) is then used to convert ***I'*** to a binary image ***BI***, which represents an edge map (i.e., 0 for background and 1 for edge). Let $\Phi\varepsilon \subset$ ***BI*** be the area corresponding to $\Phi \subset$ ***I***. $\Phi\varepsilon$ is the edge map of the source region.

After the edge map is computed, we revised the algorithm discussed in [9]. Let $\delta\Omega$ be a front contour on Ω and adjacent to Φ. We compute the initial Confidence Value of each pixel in ***I***, which is due to [9]:

$$C(p) = 1 \textit{ iff } p \in \Phi \quad \text{and} \quad C(p) = 0 \textit{ iff } p \in \Omega$$

We use the same notation as [9]. Let Ψ_p be a patch centered at pixel p $\in \delta\Omega$. We also use the same concept to compute the confidence term C(p):

$$\forall\ p \in \delta\Omega,\ C(p) = (\textstyle\sum_{q \in (\Psi p \cap \Phi)} C(q)) / |\Psi_p|$$

where $|\Psi_p|$ is the area of Ψ_p. Essentially, the confidence term computes the percentage of useful pixels in a patch Ψ_p. Useful pixels mean the coverage of source pixels in Φ. These useful pixels are used in the original algorithm proposed in [9] to find a best matching patch in the source area.

Next, we discuss the main differences of our newly proposed algorithm. The data term proposed in [9] uses an isophote and a unit vector. Data term D(p) seems to favor in-pointing areas in Ω. But, the confidence term C(p) seems to favor out-pointing areas. We propose another approach which revises the data term, based on the observation on continuous structure derivation, which we think of as the most important factor. Our assumptions follow. First, the confidence term summarizes the percentage of useful pixels in a patch. Structural information is not included. For instance, a vertical line and a horizontal line in two different patches (with the same number of useful pixels) could have the same confidence term. We believe the use of confidence term only indicates how much information in a target patch can be used as a reference while we look for similar patches in the source area. This also derives our second assumption that a better matching strategy should be used (i.e., *patch template*).

Instead of computing the isophote, we compute the percentage of edge information contained in the patch, by obtaining information from the edge map of the source region (i.e., $\Phi\varepsilon$). In addition, color variation in the patch is used:

$$\forall\ p \in \delta\Omega,\ D(p) = min(1, (\textstyle\sum_{q \in (\Psi p \cap \Phi\varepsilon)} c)) * var(\Psi_p) / |\Psi_p|$$

$$var(P_{i,j}) = \frac{\sqrt{\sum_{\forall i}\sum_{\forall j}(x_{ij}-\bar{x})^2}}{i\times j-1}, \quad \bar{x} = \frac{\sum_{\forall i}\sum_{\forall j} x_{ij}}{i\times j}$$

Note that, the constant, *c*, represents the weight (set to 1 in our experiment) of a pixel. The priority map is computed the same as [1]. Conclusively, we have

$$P(p) = C(p)*D(p), \text{ with } D(p) \text{ revised.}$$

Next, *patch template* introduces an accurate search. Let $\Psi_{p^\wedge}$ be a patch of the highest priority. This can be formulated as:

$$\Psi_{p^\wedge} = max(\forall\Psi_p,\ p \in \delta\Omega)$$

In general, we believe the larger the patch, the better chance to find a good match. Thus, it is necessary to consider surrounding useful pixels in addition to only using the useful pixels within a patch. Let $\Gamma_{p^\wedge}$ be a patch template of $\Psi_{p^\wedge}$:

$$\Gamma_{p^\wedge} = \cup_{\pm k\Psi p^\wedge}(\Psi_{p^\wedge} \cap \Phi) \neq \{\}, \text{ where } \{\} \text{ is an empty set.}$$

where $\pm k\Psi_{p^\wedge}$ represents the patch $\Psi_{p^\wedge}$ plus its surrounding patches within a distance of *k*. Our experiment sets *k* = 1; thus the eight neighbor patches are used. Note that, the intersection with the source region Φ is computed to ensure that no "empty patch" is used. With the patch template defined, the search procedure can be discussed. Let $\Gamma_{q^\wedge}$ be the best matching patch template against all candidate patch templates in the source image:

$$\Gamma_{q^\wedge} = min_{\ \Gamma q \in \Phi qr}\ d(\Gamma_{p^\wedge}, \Gamma_q)$$

where $\Phi_q^{\ r} \subset \Phi$ represents a region of source image centered at q by a distance of *r* pixels. Note that, the search iteration looks for patch templates in pixel-by-pixel steps. We define $d(\Gamma_{p^\wedge}, \Gamma_q)$ as a distance function of two patch templates:

$$d(\Gamma_{p^\wedge}, \Gamma_q) = SSD_{\ CIE\ Lab}\ (\Gamma_{p^\wedge}, \Gamma_q) * min(1, (\textstyle\sum_{q \in (\Gamma q\ \cap\Phi\varepsilon)} c))$$

The distance function considers two factors: the SSD and the number of useful pixels in the patch template. We want to find a best patch $\Psi_{q^\wedge}$ where $\Psi_{q^\wedge} \subset \Gamma_{q^\wedge}$. Hence the inpainting algorithm copy $\Psi_{q^\wedge}$ to $\Psi_{p^\wedge}$, $\forall p \in \Psi_{p^\wedge} \cap \Omega$ (only copy pixels to the empty positions in patch $\Psi_{p^\wedge}$, without changing the source area). The constant, *c*, represents the weight of a useful pixel. Usually, it is set to 1. The last step in the inpainting algorithm is to update the confidence map C(p). In [9],

$$C(p) = C(\hat{p}),\ \forall p \in \Psi_{p^\wedge} \cap \Omega$$

Essentially, a copied pixel, p, has its confidence map updated by the confidence term computed from the patch $\Psi_{p^\wedge}$. That is, all copied pixels have the same confidence value, which reveals the percentage of useful pixels in $\Psi_{p^\wedge}$. We revise the strategy as

$$C(p) = C(\hat{p}) * (d\ (\Psi_{p^\wedge}, \Psi_{q^\wedge}) / \alpha),\ \forall p \in \Psi_{p^\wedge} \cap \Omega$$

where α is a normalization factor such that (d ($\Psi_{p^\wedge}$, $\Psi_{q^\wedge}$) / α) is in (0, 1]. Thus, we incorporate the degree of similarity into the confidence map. Note that, the same distance function for two patch templates can be applied to two patches.

However, because of both terms we mentioned above contains no structure information, we still incorporate a structure reconstruction mechanism into our inpainting procedure. This mechanism will be discussed in Section III.

C. *Bézier curves*

Bézier curve is an efficient method for designing polynomial curves when you want to control their shape, outline or pattern module with an easy way. Many CAD works using Bézier curves to design their shape, outline or pattern module.

The appearance of these objects in the real world is much similar to Bézier curve. When we want to reconstruct the contour lines, Bézier curve is the useful method. To describe how the path of the Bézier curve goes which depends on the given control points.

Given the called control points $\boldsymbol{P}_{CC}=\{\boldsymbol{P}_0, \boldsymbol{P}_1, \boldsymbol{P}_i..., \boldsymbol{P}_n\}$, $0 \le i \le n$ for the Bézier curve. The general form of Bézier curve of degree n are

$$B(t)=\sum_{i=0}^{n}\binom{n}{i}P_i(1-t)^{n-i}t^i$$
$$=P_0(1-t)^n+\binom{n}{1}P_1(1-t)^{n-1}t+...+P_nt^n \ ,$$
$$t\in[0,1]$$

where t is the variable of time.

Bézier curve can be in any degree, in this paper, we only use three control points to describe a Bézier curve, which is a quadratic Bézier curve. Figure 4 shows two quadratic Bézier curves. The general quadratic formula of Bézier curve is given as following:

$$B(t)=(1-t)^2P_0+2t(1-t)P_1+t^2P_2, t\in[0,1]$$

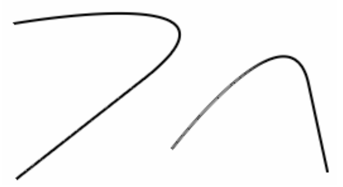


Figure 4. Two quadratic Bézier curve

III. Proposed Algorithm

The strategy of our algorithm is simple and efficient, find the curves, extend the curves, fit the curves. We use the curvature of a contour line to determine the length of contour line should be taken as the experimental template. Figure 5 shows the vectors on contour line.

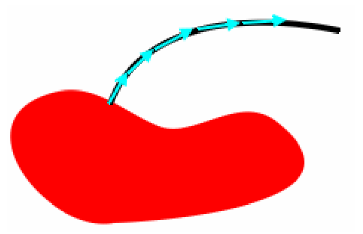


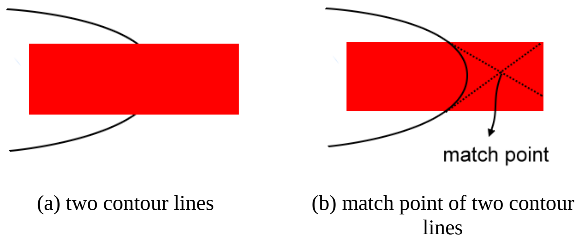
Figure 5. Vectors on contour line

This is a basic algorithm for *n vs. n points on both sides* of the inpainting domain which can be described as following

1. Define the inpainting domain D.
2. For every contour lines L_i over the boundary ΩD of D, find point $C_{i,1}$ on ΩD.
3. Starting from $C_{i,1}$, find $C_{i,j}$ on the same contour line, where $C_{i,j+1} = C_{i,j} + \Delta t$, Δt is the Euclidean distance between each point $C_{i,j}$, $j = 2\sim5$
4. For every two $C_{i,j}$ points, we can calculate the difference, we use the structure of vector to represent the difference. $v[k] = C_{i,k+1} - C_{i,ki}$
5. Calculate the cosine of angel between $v[k]$ and $v[k+1]$, if the value is greater than Δa, $C_{i,k}$ is a point we should stop there.
6. To computing the variation of the slope of the contour line according the

$$S_i = C_{i,j} + 0.5*(C_{i,j}-C_{i,j+1}) + 0.25*(C_{i,j+1}-C_{i,j+2}) + 0.25*(C_{i,j+2}-C_{i,j+3})$$

7. For every point $C_{i,1}$ on one side of ΩD, there is a corresponding point on the other side of ΩD, we use these 2 points and their estimated slope to extend their contour line using Bresenham line-drawing algorithm.
8. There is a match point as shown as Figure 6, and two corresponding points. We use these three points to render the Bézier curve.
9. Repeat step 3 to 8, each n to n contour lines are mapped and reconstructed.



(a) two contour lines (b) match point of two contour lines

Figure 6. Contour lines

We have a basic *n to n* contour line, now we can extend it to *n+1 to. n*. We can reduce two points on *n+1* side and one point on *n* side by determine which point should connect with which two points on the other side.

The following is how we decide which 3 points should link together.

In the case of "4" to "3" points in Figure 7, there are three probabilities including "a1-b1-a2", "a2-b2-a3", and "a3-b3-a4". Then we take a look at each estimated slope and the direction of their contour line. For example, if the direction of a2 goes to left and a3 is to right, they won't link together.

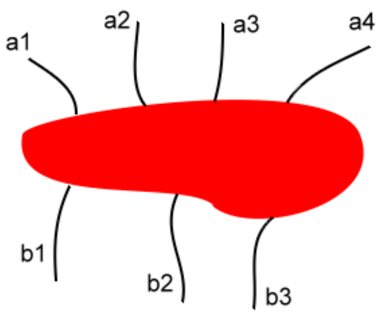


Figure 7. a4 to a3 contour lines domain

After all of missing contour lines are reconstructed, we will apply our inpainting procedure to recover damaged area.

IV. EXPERIMENT RESULTS

The proposed algorithm has been test on several kinds of images. Several parameters which used in our image inpainting procedure are summarized below with values:

1. The size of patches is set to 5 by 5 for all images.
2. The distance for searching patches is set to 15.
3. index for patch template is set to *1*

Figure 8 to Figure 10 show results of curve recovering respectively. In each figure, (a) is the original image, (b) shows the damaged image, (c) is the result of contour recovering, (d) is the inpainting result.

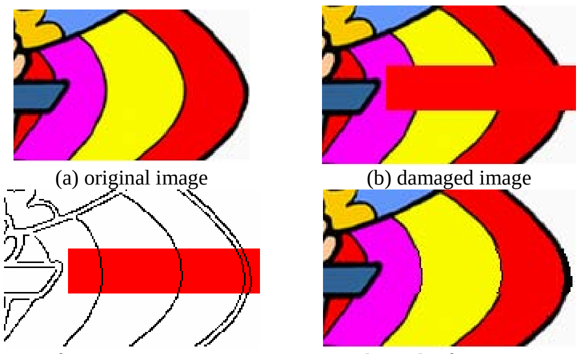


Figure 8. Curve Repairing

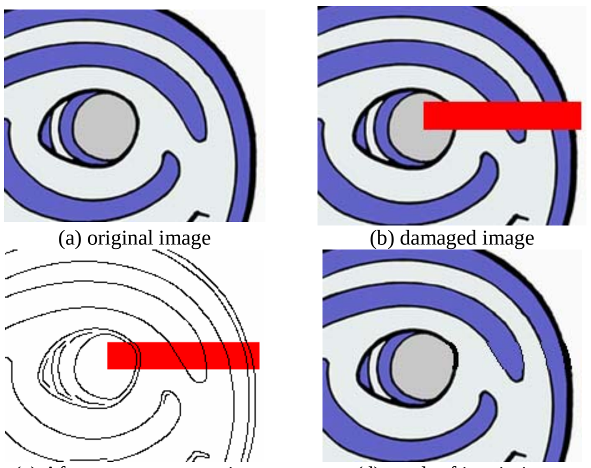


Figure 9. Curve Repairing

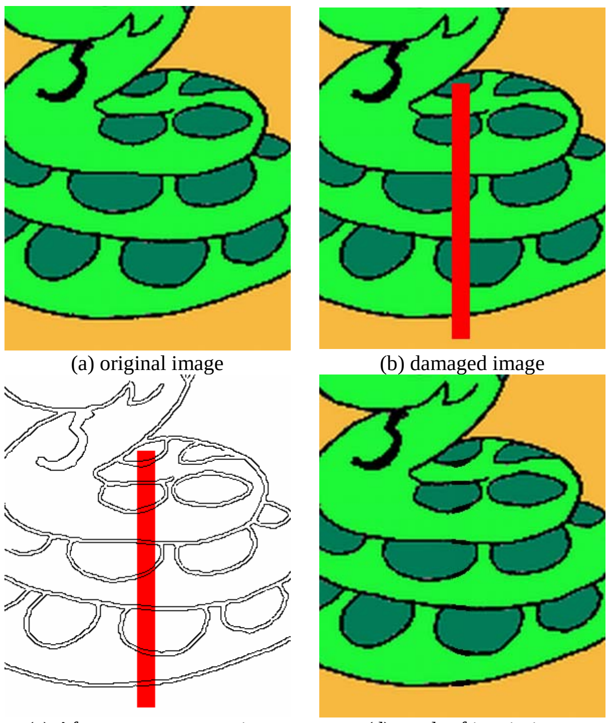
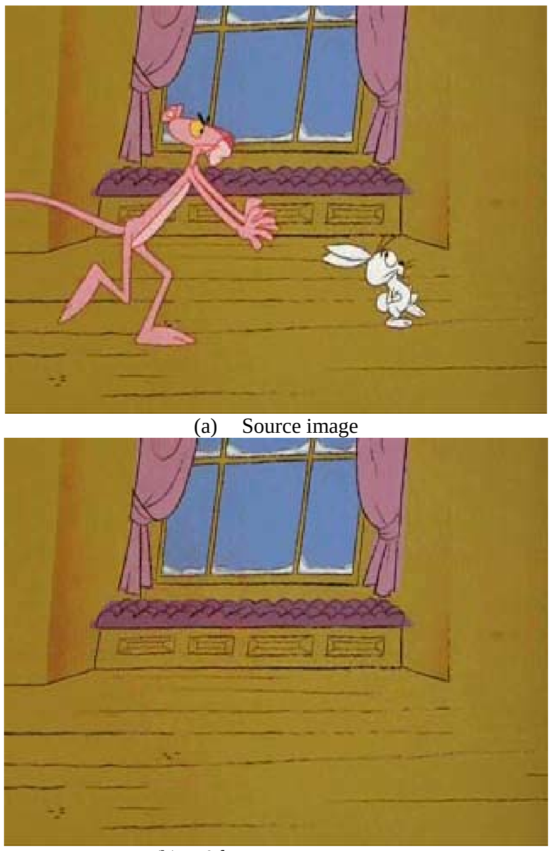


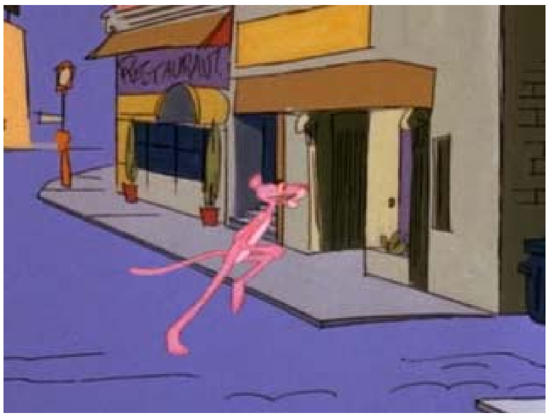
Figure 10. Cartoon Inpainting - Snake

Figure 11 to 14 shows the result that we apply our image inpainting procedure to realize object removal in several kinds of image.

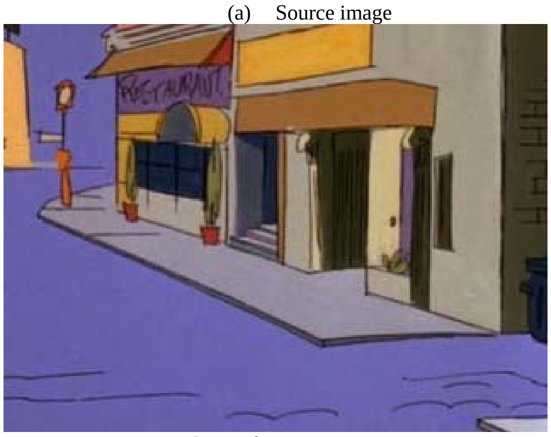


(b) After Image Inpainting

Figure 11. Image Inpainting - Cartoon

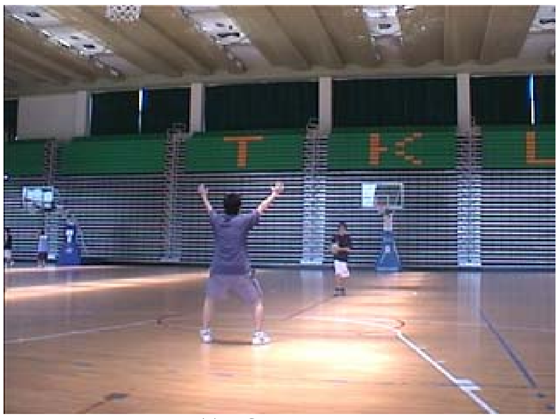


(a) Source image

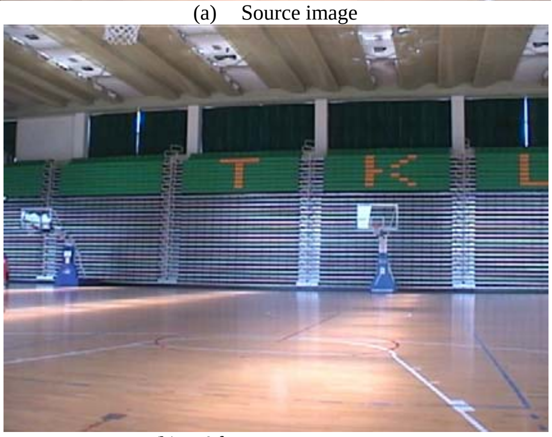


(b) After Image Inpainting

Figure 12. Image Inpainting - Catroon

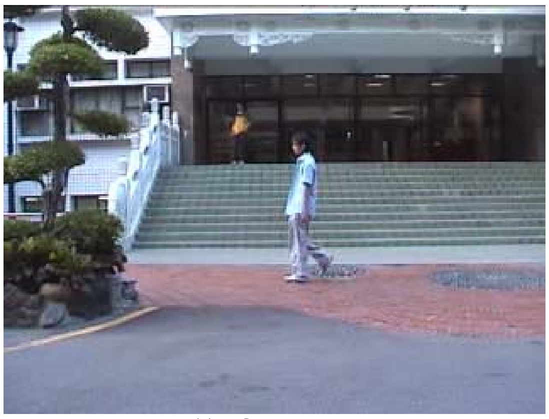


(a) Source image

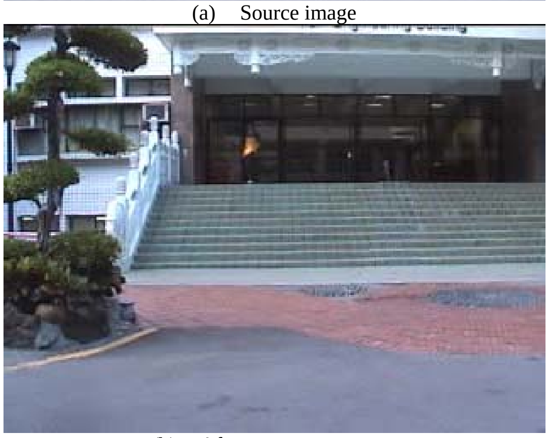


(b) After Image Inpainting

Figure 14. Image Inpainting – Basketball Court



(a) Source image



(b) After Image Inpainting

Figure 13. Image Inpainting - Stairs

We also provide an offset table to show the difference among the reconstruct contour line by our algorithm and straight line across inpainting domain and the original contour map. The offset is calculated by how many pixels away from the original contour map at some sample levels show in figure 15. We show the average of offsets in table 1.

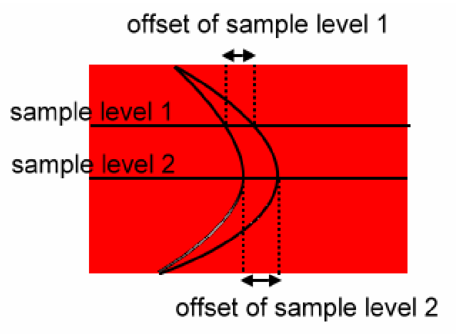


Figure 15. several offsets of sample levels

TABEL 1.
THE AVERAGE OFFSET OF FIGURE 14.

line # / method	1	2	3	4	5	6	7
straight line	2.25	2	1.5	1.25	0.75	0.5	0.5
proposed algorithm	0.45	0.55	0.5	0.25	0.25	0.45	0.45

V. CONCLUSION AND FUTURE WORKS

We proposed a novel algorithm for inpainting images based on the contour lines repairing and exemplar-based image inpainting. Firstly, we will using mean shift segmentation to realize color segmentation in damaged image. Secondly, Bézier curve is applied to connect the missing contour lines to reconstruct main structure in damaged area. Finally, we use our inpainting procedure to find a best matching patch from other source region which contains "real" information and pasted it into corresponding position.

We will try to use our inpainting procedure on other kinds of images (as shown in Figure 16 and 17) which contains complicated structure information in image to complete work of object removal.

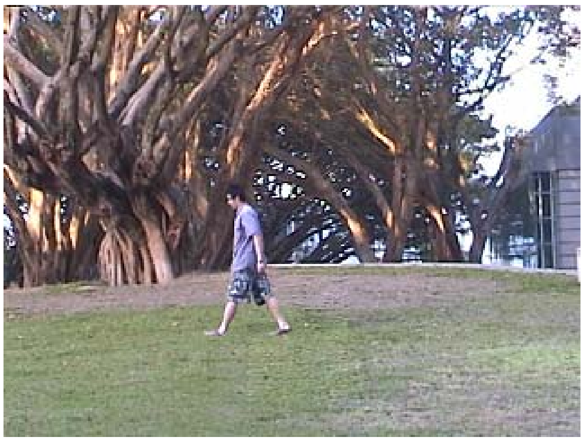


Figure 16. Image with complicated structure information - Park

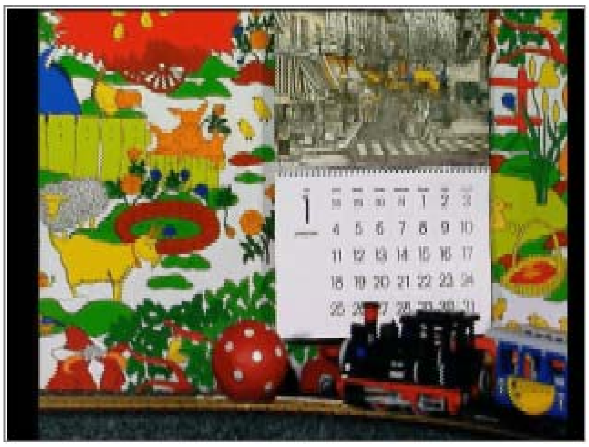


Figure 17. Image with complicated structure information - Train

REFERENCES

[1] M. Bertalmio, G. Sapiro, V. Caselles, C. Ballester, *Image Inpainting*. Proceedings of SIGGRAPH 2000, pages 417-424.

[2] Shih-Chang Hsia., *A fast efficient restoration algorithm for high-noise image filtering with adaptive approach*, Journal of Visual Communication and Image Presentation, Pages 379-392, June, 2005.

[3] T. Chan, J. Shen, *Mathematical Models for Local Deterministic Inpaintings*, Technical Report, CAM00-11, Image Processing Research Group, UCLA,2000

[4] T. Chan and J. Shen. *Non-Texture Inpainting by Curvature Driven Diusions (CDD)*, Technical Report, CAM00-35, Image Processing Research Group, UCLA,2000

[5] M. Oliveira, B. Bowen, R. McKenna, and Y.-S. Chang. *Fast Digital Image Inpainting*, Proceedings of VIIP 2001,pages 261-266

[6] D. Comaniciu, P. Meer, *Robust analysis of feature spaces: Color image segmentation*. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, June 1997, 750-755.

[7] P. Meer, B. Georgescu, *Edge detection with embedded confidence*, IEEE Trans. Pattern Anal. Machine Intell, vol. 23, 1351-1365, 2001.

[8] Wikipedia - Bézier curve. http://en.wikipedia.org/wiki/Bezier_curve

[9] A. Criminisi, P. Perez and K. Toyama, *Region Filling and Object Removal by Exemplar-Based Image Inpainting*, *IEEE Trans. on Image Processing*, vol. 13, Sept. 2004, pp. 1200-1212.

[10] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion," *ACM Trans. Graphics (SIGGRAPH)*, vol. 22, San Diego, CA, pp. 303-312, 2003.

[11] J. Hays, Alexei A. Efros, "Scene Completion Using Millions of Photographs," *ACM Trans. on Graphics (SIGGRAPH 2007)*, vol. 26, No. 3, August 2007.

Jason C. Hung is an Assistant Professor of Department of Information Technology, The Overseas Chinese Institute of Technology, Taiwan, R.O.C. His research interests include Multimedia Computing and Networking, Distance Learning, E-Commerce, and Agent Technology. From 1999 to date, he was a part time faculty of the Computer Science and Information Engineering Department at Tamkang University. Dr. Hung received his BS and MS degrees in Computer Science and Information Engineering from Tamkang University, in 1996 and 1998, respectively. He also received his Ph.D. in Computer Science and Information Engineering from Tamkang University in 2001. Dr. Hung participated in many international academic activities, including the organization of many international conferences. He is the founder and Workshop chair of International Workshop on Mobile Systems, E-commerce, and Agent Technology. He is also the Associate Editor of the International Journal of Distance Education Technologies, published by Idea Group Publishing, USA. The contact address of Dr. Hung is Department of Information Technology, The Overseas Chinese Institute of Technology, No:100, Chiao Kwang Rd., Taichung 407, Taiwan (email:jhung@ocit.edu.tw)

Yi-Chun Liao is an assistant professor of the Department of Computer Science and Information Engineering at China University of Technology, Taiwan. He received the PhD degree from Tamkang University, Taipei, Taiwan in 2005, and received his BS and MS degrees in Computer Engineering from Tamkang University in 1998 and 2001, respectively. His current research interests include Multimedia Computing, e-Learning, and Video Processing.

Chun-Hong Huang is an assistant Professor of the Department of Computer Information and Network Engineering at Lunghwa University of Science and Technology. His current research interests are in the areas of multimedia processing, 3D information analysis and retrieval. His contact email is ch.huang@mail.lhu.edu.tw.

Nick C. Tang received the PhD degree from Tamkang University, Taipei, Taiwan in 2008, and received his BS and MS degrees in Computer Engineering from Tamkang University, respectively. His current research interests include Multimedia Computing, and Video Processing.

Ta-Jen Chen is a MS student graduated from the Department of Computer Science and Information Engineering at Tamkang University, Taiwan. He received the BS and MS degrees from the same university in 2005 and 2007, respectively. His research interests includes computer vision, video analysis, and their applications.