行政院國家科學委員會補助專題研究計畫 √ 成 果 報 告 □期中進度報告

（計畫名稱）

計畫類別：√ 個別型計畫　　□ 整合型計畫
計畫編號：NSC 94-2213-E-032 -017－
執行期間：94 年 08 月 01 日至　95 年 07 月 31 日

計畫主持人：施國琛

計畫參與人員： 林宏、葉瑋松、何宗達

成果報告類型(依經費核定清單規定繳交)：√精簡報告　□完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：淡江大學資訊工程學系

中　華　民　國　95　年　12　月　15　日

影像瑕疵偵測與修補技術之研究

Photo defect detection and inpainting

計畫編號： NSC 94-2213-E-032 -017
執行期限： 94/08/01 ～ 95/07/31

計畫主持人：施國琛

ABSTRACT

Image inpainting (or image completion) techniques use textural or structural information to repair or fill damaged potions of a picture. However, most techniques request a human to identify the portion to be inpainted. We developed a new mechanism which can automatically detect defect portions in a photo, including damages by color ink spray and scratch drawing. The mechanism is based on several filters and structural information of damages. Old photos from the author's family are used for testing. Preliminary results show that most damages can be automatically detected without human involvement. The mechanism is integrated with our inpainting algorithms to complete a fully automatic photo defects repairing system.

Key words: image inpainting, defect detection, structural features, image processing, image restoration

1. INTRODUCTION

The discussion of image and video inpainting were found in the literature. In most image inpainting projects, users have to select the defect area. On the other hand, since it is impossible to select defects on all video frames, video inpainting uses motion estimation, shape features, and Kokaram's model [1] to detect defects such as line scratch and spike on video. Unfortunately, motion estimation and Kokaram's models can not be used in defect detection of a single photo image. Thus, we propose a naïve but effective algorithm to detect photo defects. The detection of defects is similar but different from color image segmentation [5] to find fore ground objects. The detection is also different from finding defects in random color textures [4], using color cluster scheme. Basically, ink sprayed on image will destroy the homogeneity of a photo. Thus, the detection mechanism relies on the property of ink (e.g., ink color, shape, and decadence of gradient). Our mechanism can further distinguish the two types of damages (ink spray and scratch) in different ink colors. The proposed algorithms contain several filters, which will be discussed in the next section. The experimental results include more than 40 damaged photos. Some old photos are collected from the author's family history, with real cases of damages. A few photos are new photos with real ink spray and scratches scanned for testing.

2. THE DEFECT DETECTION ALGORITHM

2.1 *The Defect Intensity Detector*

We propose an algorithm which automatically adjusts threshold values to achieve the best results of detection. We use HSI color space since it is closer to human perception. It has been discussed that RGB color space is not quite suitable for comparing image colors due to the discontinuity of color values. Our experiments on different color spaces also reveal that HSI color space can be used in several successful examples.

In this paper, we aim at handling the ink spray defects, with an extension to detect scratch by color pens. There are plenty type of ink colors, and photo images usually contains lots of different colors. Hence, if we try to detect ink with color information, it would be almost impossible. We consider the intensity and shape of photo defects. Though, it is almost not possible to discriminate ink regions from objects in photo images. The intensity variation of ink regions is usually quite smooth and steady while comparing with objects in photos. Thus, we use the HSI color space and use intensity in the first filter. We decrease the value of I (i.e., intensity) from 255 to 0 at each step. In each step, we record the number of pixels detected. It has been mentioned that the intensity variation of ink is usually low. In decreasing the intensity of a photo, the ink part will remain while other parts will be filtered, up to an estimated I value. In the second filter, we compute a pixel histogram, with pixel numbers on y-axis and adjustment of I values on x-axis. That means we record the number of pixels been detected in each I variation. The histogram is represented as a curve chart in our tool (see Figure 1). We calculate the variance of pixel number been detected between two different steps of adjustment. If the variance is low, it means that pixels been detected in the last adjustment are not much affected by the present adjustment of I. And the pixels been detected are possible to be pixels of ink spray because of its steady intensity. Keeping on the calculation of variance of pixels been detected, a collection consists of consecutive adjustments of I can be constructed. The collection needs to be analyzed since sometimes the variance of pixels been detected is low only because it has low discrimination in a step of I adjustment. Then the collection been record has no use and we can't detect the ink. Thus, if the collection of adjustment contains too

many passes of adjustment, we just leave it and go on to analysis the photo image. After the calculation and analysis, a set of adjustment of I can be found out to separate pixels of ink from objects in the photo. The following is our algorithm to find the possible minimal intensity of a defect area.

Algorithm*: *Defect Intensity Detector

Let $\boldsymbol{\lambda}$ be a maximum length of a continuous curve ($\lambda \geq 5$)
Let $\boldsymbol{\alpha}$ be a threshold of difference of pixel number (initially, $\boldsymbol{\alpha} = 100$)
Let n be a value of pixel number ($n = 100$)
Input: Image C
Output: Intensity $\boldsymbol{\delta}$
Algorithm:
Step1: Convert image C to HSI color space
Step2: Compute the number of pixels, w. r. t. each I variation; Let $\#P_i$ represents the number of pixels in the i_{th} I variation, where $1 \leq i \leq 255$
Step3: Compute $\Delta P_i = (\#P_i - \#P_{i-1})$, if $\Delta P_i < \boldsymbol{\alpha}$ store i in an index list L
Step4: Search L and store the consecutive elements in set S, $S_i = \{\boldsymbol{\delta}, M_i\} \in S$; where $\boldsymbol{\delta}$ is the start intensity of consecutive elements, and M_i is the length of the i_{th} consecutive elements
Step5: Find S_i with the maximal length M_i in the set S, where the consecutive elements have M_i less than or equal to $\boldsymbol{\lambda}$
Step6: If no $M_i < \boldsymbol{\lambda}$ then $\boldsymbol{\alpha}=\boldsymbol{\alpha}+n$; goto **Step3**
else Return $\boldsymbol{\delta}$

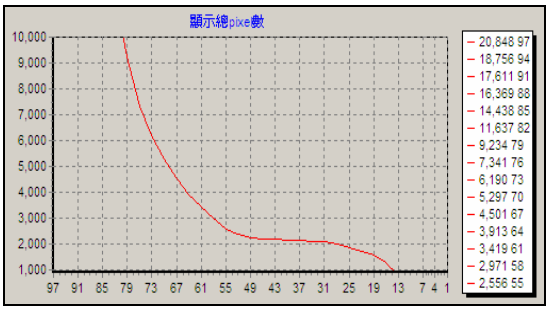


Figure 1: Histogram of Pixel Numbers by Intensity

2.2 *The Adaptive Filter*

The *defect intensity detector* operation is not ready to separate defect object and background object yet. We use an *adaptive filter,* which considers the details of object properties, and decide which part is defect. The intensity found in the defect intensity detector can be used on finding the potential defect objects of a photo. The *adaptive filter* will detect multiple objects. Each object O_j in an object set O has two parameters, A_j and B_j, to represent object size and object boundary, respectively. We define a threshold of object size, $\boldsymbol{\mu}$. If an object size is small than the $\boldsymbol{\mu}$ then we remove the object from set O. Two morphological operations, *opening* and *closing*, are then used. The *median filter* operation with 3×3 filter windows performs isolate object detection and elimination.

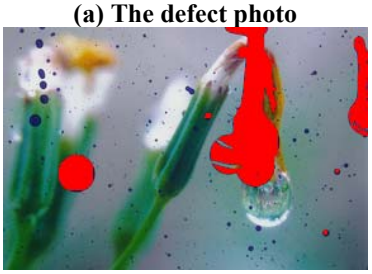
In the *adaptive filter* algorithm, the adjustment of I is achieved by decreasing the value of I from $\boldsymbol{\delta}$ to carry out the defect object to be detect by an intensity value of 5 at each step. In each step of adjustment, we record the size and locations of object O_j detected in a new set Q. The parameter n is a number of the objects in set Q. The new set Q is for intensity iteration while the original set O is used as a reference. In intensity iteration, an object can be split. The size of object (includes split objects) can be reduced. However, the center of the object is not likely to be altered. To tell whether split objects belong to an original object, we compare the center of split object against the boundary of the original object. It is important to keep track of which split object belong to an original object since we calculate the decreasing rate of area. If the decreasing rate is high, the object is not likely to be a defect. After several intensity iterations, the object left in Q are defects. We mark the original defect objects in O on the photo. The *Adaptive Filter* algorithm follows.

Algorithm*: *Adaptive Filter
Let O be a set of objects; $O_j = \{A_j, B_j\} \in O$, where A_j and B_j are the area size and boundary of object O_j, respectively
Let $\boldsymbol{\Omega}$ be a percentage threshold of object size ($\boldsymbol{\Omega} = 50\%$)
Let $\boldsymbol{\mu}$ be a threshold of number of pixels ($\boldsymbol{\mu}$= 5 or 10)
Let $\boldsymbol{\varepsilon}$ be a decreasing intensity amount ($\boldsymbol{\varepsilon} = 10$)
Algorithm:
Step1: Store initial objects obtained from color segmentation in set O
Step2: Remove small objects form set O if the area of object size $A_j < \boldsymbol{\mu}$
Step3: Use *filter Opening*(*Closing*(C)) to remove isolated objects in O
Step4: Copy set O to a new set Q; Let $\boldsymbol{n}$ be the number of objects in Q; Let $s = \boldsymbol{\delta}$ be an initial intensity value (from the defect intensity detector)
Step5: Intensity Iteration: Repeat till $\boldsymbol{n}$ has no change
{
For all object O_i in Q do
If A_i decreases > $\boldsymbol{\Omega}$ from last Intensity Iteration and O_i is not split then
remove O_i from Q and update $\boldsymbol{n}$
Set $s = s - \boldsymbol{\varepsilon}$
}
Step6: For each element O_j in set O not removed in Q
Mark O_j in image C

3. EXPERIMENTS RESULTS AND DISCUSSION

In this section, we present the experimental results from a variety of tests. We divide the photos into two categories, with damages by spray of color ink (blue, red or black) and scratch by ink pen. Examples of defect pictures and detections are presented in Figure 2 to 6. Figure 2 shows a defect photo by blue ink. In this experiment, we set the threshold of object size μ to 10 pixels. Therefore, there are some small objects not detected. Figure 3 has defects by black ink. The threshold of object size μ is also 10 in this case. Note that, the strings in black of figure 3 (a) is not changed mostly.

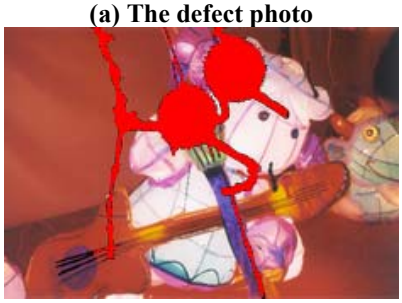
(a) The defect photo



(b) The detection result

Figure 2: Detection of Defect with Blue Ink Spray

(a) The defect photo



(b) The detection result

Figure 3: Detection of Defect with Black Ink Spray

(a) The original picture **(b) The detection result**

Figure 4: Defect Detection with Ink Pen Scratch (some thin line scratches are not detected)

A simple example of scratching defect is shows in Figure 4. Scratch lines are hard to detect, especially when the line is very thin. In figure 4, we take an alternative strategy, which use gray levels of image. The gray value detected is 128. Table 1 summarizes parameters used in our algorithm for figures 2 to 6.

Table 1: Parameters for Defect Detection

Figures	δ	α	μ	Gray value
2	46	300	10	
3	37	100	10	
4		100	5	128
5(a)	43	100	10	
5(b)	19	100	10	
5(c)	28	100	10	
5(d)	26	100	10	
6(a)	50/37	200	10	
6(b)	50/40	100	10	
6(c)	60/22	200	10	

We also consider the type of defects. A naive but effective strategy is to compute the ratio of area to perimeter on defect objects. We use a line detection algorithm to find the approximation of pixel numbers of perimeters. Roughly speaking, ink sprays will have a ratio higher than line scratches due to shape. In figure 5, we illustrate four sets of photos with type of defects separated. We found that the ratio of ink spray is usually higher than 1 and the ratio of line scratch is close to 1. However, small ink spray with a diameter close to line thickness is detected as a line. A threshold set to 2.0 is used to separate the two types of defects, using red to show ink spray and blue to show line scratch. Table 2 shows the ratios of defect objects in figure 5. Each photo contains from 2 to 3 defect

objects. The numbers of pixels in object area (i.e., O_A) and object perimeter (i.e., O_P) are also provided.

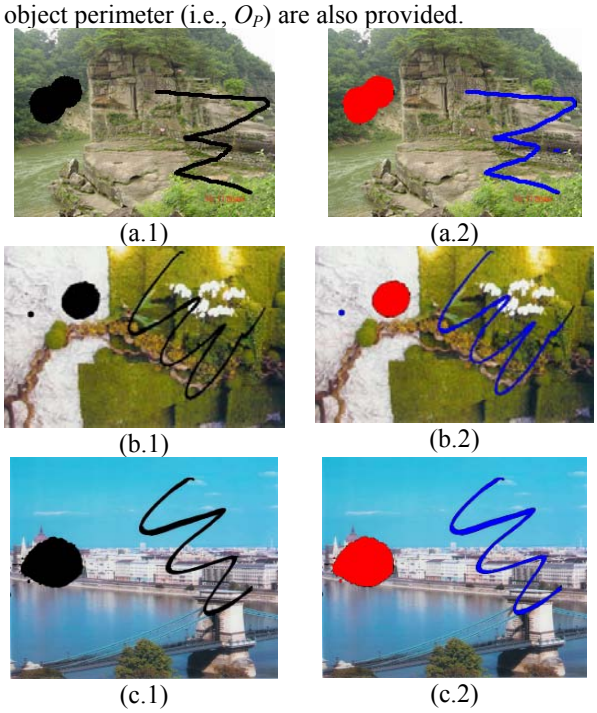


Figure 5: Separation of Defect Types

Table 2: Ratio of Pixel Numbers

Fig.5	(a)		(b)			(c)	
O_{item}	#1	#2	#1	#2	#3	#1	#2
O_A	2252	2013	1915	2029	48	1636	2902
O_P	411	1953	2744	354	48	2085	479
Ratio	5.47	1.03	0.69	5.73	1	0.78	6.05

We also detect multiple ink colors in the same photo. The strategy detects two different intensity values. The values detected for blue/black are 50/37 (6a), 50/40 (6b), and 60/22 (6c). As shown in figure 6, the black ink detected is shown in red and the blue ink detected is shown in yellow.

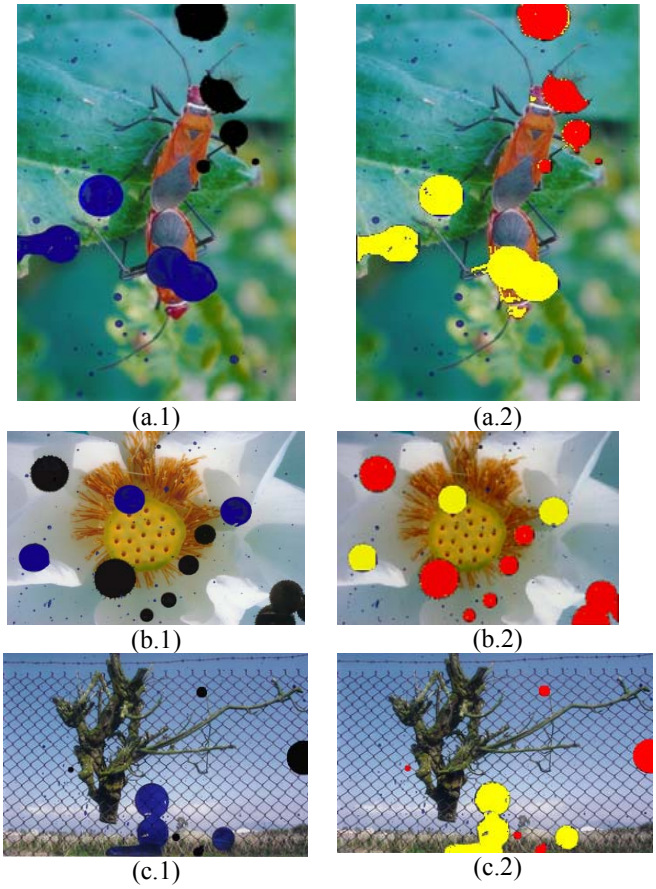


Figure 6: Separation of Defect Ink Colors

4. IMAGE INPAINTING MOTHED

The defect inpainting strategy we used was proposed in [6,7]. We briefly summarize the multi-resolution inpainting strategy here. Considering an image or a video frame as a large Damaged Image Block (DIB), as shown in Figure 8, a DIB can be subdivided into several Image Blocks (IBs). Each Image Block may or may not contain damaged pixels. Furthermore, each Image Block is subdivided into several Pixel Blocks (PBs). Pixel Blocks are elementary objects to be inpainted. We believe pixel color variance has a strong indication of the degree of details in a block. The threshold **α** sets the criterion of whether a multi-resolution inpainting is required. In our implementation, the value of **α** is a percentage in the range between 0 and 100 (the maximum *var*) of an **IB**. Another criterion is the percentage of potential damaged pixels. We argue that, if the percentage 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 (see Figure 7) iterates through each of the **IB**s in a **DIB**. If the color

variance of IB is below the threshold **α**, there is not much difference of pixels in **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 β_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 β_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 **α**, the algorithm is called recursively to handle the next level of details. In the article [6], a complete analysis is given to show the inpainted results are satisfiable. There are three thresholds in the algorithm, α, β_1 and β_2. We use $\alpha = 80$, $\beta_1 = 85$, and $\beta_2 = 95$. The experiments to obtain these values are discussed in [6].

```
Algorithm inPaint(block DIB)
  if DIB is a small block then return
   for each image block IB
     if var < α then
     {
      for each PB in the image block
      {
       if the percentage of damaged pixels in PB > β2
          inpaint the damaged pixels using Mcolor
       else
          if the percentage of damaged pixels in PB > β1
             inpaint the damaged pixels using Ncolor
          else
             inpaint the damaged pixels using neighbor pixels
      }
      for each pixel in the boundary of each PB
          smooth boundary pixels using neighbor pixels
      }
     else
                call inPaint(IB)
```

Figure 7. The inpainting algorithm

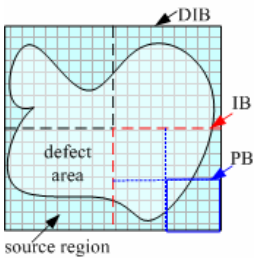


Figure 8. Damaged Image Block (DIB), Image Block (IB) and Pixel Block (PB)

5. PRELIMINARY EXAMPLES

Some test results of defect photo inpainting are shown in Figure 9 and Figure 10. Different categories of real photo are tested on different types of defects.

(a) The Defect Photo by Ink

(b) The Detected Result

(c) The Inpainting Result

Figure 9. Inpainting Result of Defect Photo with Black Ink Spray

(a) The Defect Photo by Scratch

(b) The Detected Result (some thin line scratches are not detected)

(c) The Inpainting Result

Figure 10. Inpainting Result of Defect Photo with Line Scratches

6. CONCLUSIONS

Digital image inpainting algorithms usually rely on user to select defect areas to be inpainted. We propose a naive but effective strategy to detect ink spray and line scratch with different ink colors. We found that, on different type of photographic papers, ink spray results in different types of intensity detection. Also, thickness of ink results differently. In addition, if scratch lines are very thin, it is hard to detect. In such case, we may subdivide the photo into multiple segments for separated detection. Our future research will detect other types of defects, such as ripped and frazzled photos.

REFERENCE

[1] Vittoria Bruni and Domenico Vitulano, “A generalized model for scratch detection,” IEEE Transactions on Image Procesing, Vol. 13. No. 1, January 2004, pp. 44-50.

[2] L. Joyeux, O. Buisson, B. Besserer, S. Boukir, "Detection and removal of line scratches in motion picture films,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 23-25, 1999, pp. 548-553.

[3] A. Machi, F. Collura, “Accurate spatio-temporal restoration of compact single frame defects in aged motion picture,” on Proceedings 12th International Conference on Image Analysis and Processing, 2003., 17-19 Sept. 2003, pp. 454-459.

[4] K. Y. Song, J. Kittler and M. Petrou**, “**Defect detection in random colour textures**,”** Image and Vision Computing, vol. 14, no 9, October 1996, pp. 667-683.

[5] Swee-Seong Wong and Wee Kheng Leow, “Color segmentation and figure-ground segregation of natural images,” in Proceedings on International Conference on Image Processing, ICIP’02, 2000, pp. 120-123.

[6] Timothy K. Shih., Liang-Chen Lu, and Rong-Chi Chang, “Multi-Resolution Image Inpainting,” in Proceedings of the 2003 IEEE International Conference on Multimedia & Expo (ICME'03), July 6 – 9, 2003, pp. 485-488.

[7] Timothy K. Shih., Rong-Chi Chang, Liang-Chen Lu and Huan-Chi Huang, "Multi-layer Inpainting on Chinese Artwork," in Proceedings of the 2004 IEEE International Conference on Multimedia and Expo (ICME ‘04), 2004, pp. 21-24.