INFORMATICA, 2012, Vol. 23, No. 1, 47–63

Efficient Reconstruction of Images with Deliberately Corrupted Pixels

Bogdan LIPUŠ, Borut ŽALIK
Faculty of Electrical Engineering and Computer Science, University of Maribor
Smetanova 17, SI-2000 Maribor, Slovenia
e-mail: bogdan.lipus@uni-mb.si, zalik@uni-mb.si

Received: May 2010; accepted: October 2011

Abstract. This paper considers a new method for reconstructing deliberately-corrupted pixels in raster images. Firstly, a faster approach for reconstructing corrupted pixels is proposed by applying a processing-circle instead of a processing-square. It is shown that the obtained quality of the reconstructed image is no worse because of this. The quality of the reconstruction is further improved by controlling the pixel corrupting process within the input image. It is shown that a combination of the processing-circle approach and data-dependent corruption reduces the reconstruction time, and the mistakes of the reconstructed pixels.

Keywords: pixel corruption, pixel reconstruction, image processing and representation.

1. Introduction

This paper considers the following problem: Let's take a raster image Ω_1 defined by $n \times m$ pixels. Then, k pixels ($k < n \times m$) are deliberately corrupted. The positions (x, y) of these k pixels are known. Two possibilities are considered on how to select these k pixels. Firstly, they are determined randomly, and secondly, a data-dependent approach is applied, i.e., more pixels are corrupted within those areas where the pixel values are similar. During the reconstruction phase, an image Ω_2 of $n \times m$ pixels is obtained by calculating the values of each k corrupted pixel using the Radial-basis function. The surrounding non-corrupted pixels are used for this, in such a way that $\Omega_2 \approx \Omega_1$. Unfortunately, the reconstruction process is numerically rather intensive. The computational time depends on the number of known pixels within the neighbourhood of the corrupted pixels, because they affect the size of the system of linear equations. The amount of the neighbouring pixel's quantity is determined by the neighbourhood area, and the corruption ratio. Because of this, less non-corrupted neighbouring pixels need to be considered. The experiments showed that instead of a processing-square, as used to date, a processing-circle can be applied. In this way, the processing time is noticeably reduced without decreasing the qualities of the reconstructed pixels. It is reasonable to control the corruption process regarding image content in order to achieve better reconstruction. A simple data-dependent approach is suggested, for this purpose.

There are several applications within which the considered problem can be used. For example, images in an internet archive can be previewed when being corrupted and reconstructed after purchasing, or, images being sent through various communication channels can be accessed by an intruder. The useful values of such images are considerably smaller when deliberately corrupted.

This paper is organised as follows: Section 2 gives a brief overview of the pixel reconstruction techniques. Section 3 considers a reconstruction technique with the Radial-basis function using a processing-circle. The results are presented in Section 4 and discussed in Section 5. Finally, the paper concludes in Section 6.

2. Related Work

There are several techniques for the reconstruction of corrupted pixel values within an image. Texture synthesis techniques attempt to find those areas on an image that are similar to the area where pixels are damaged. The information from this area is then used to set the value of the wrong pixel. Efros and Leung (1999) tried to find the similar area by using a Markov random field model, and probability distribution. The new image was created outward from the initial seed using the square around the corrupted pixel. A similar technique was proposed by Bornard *et al.* (2002) and Demanet *et al.* (2003). The squared-window is used to choose the best replacement candidate from the corrupted pixel's neighbourhood. They used the normalised mean-squared distance between the similar pixels and the damaged ones. Sprott (2004) applied stochastic cellular automaton to produce fictitious fractal data that mimics the features of the actual pattern. This technique is suitable for those images that expose fractal features, such as landscapes.

The majority of the corrupted pixel reconstruction techniques propagate information from non-corrupted to corrupted areas. The corrupted region is filled with propagated information along the level lines from outside the corrupted area (Bertalmio *et al.*, 2000). As Partial Differential Equations (PDE) are used, it is difficult to achieve an implementation that is fast enough in practice. Telea (2004) and Bornemann and März (2007) used a similar approach, however they applied the marching method, which is fast and simple to implement. Their technique propagates colour information inward from the corrupted region's boundary and estimates smoothness along the image gradient using the weighted average over the known pixels. Shih *et al.* (2004) applied a colour interpolation mechanism. The pseudo squared window around the corrupted pixel is used to estimate whether there is enough information on the image to calculate the mean value. If the image is seriously damaged, the global mean value is assigned to that corrupted pixel. Recently, Wu *et al.* (2010) proposed a novel exemplar-based image completion model. They used bidirectional diffusion PDE. Their experiments showed that they could properly reconstruct the target region whilst preserving the geometrical structure within the image. Some techniques combine structured region propagation with texture synthesising (Criminisi *et al.*, 2004). Firstly, the patch-priorities are calculated by using the best-filling strategy. Secondly, the texture and structure information are propagated to find the most similar patch.

Finally, the confidence values are updated. These values are used during the first step for priority determination. As reported by the authors, the algorithm is very efficient and accurate for the synthesis of texture, and the propagation of a linear structure. Chen and Reiter (2007a) and Chen *et al.* (2007b) presented an improvement of the previous approach. The efficiency and effectiveness of the exemplar-based method were improved by a better search-strategy. Yamauchi *et al.* (2003) combined texture synthesis and image inpainting. The image is decomposed into low and high-frequency parts. The texture synthesis is used for the high-frequency part, and the fast image inpainting using discrete cosine transformation for the low-frequency part.

Another type of image reconstruction technique interpolates the values of the known pixels within the neighbourhood of the corrupted pixel. Oliveira *et al.* (2001) used convolution with a diffusion kernel. They applied the Gaussian kernel to calculate the weighted average of a pixel's neighbourhoods. This method requires anisotropic diffusion to handle the high-contrast edges. Several approaches use the Radial-basis function for determining the corrupted pixel from the known surrounding pixels. These approaches require a solving of the system of linear algebraic equations that can be done in $O(n^3)$ time. Savchenko *et al.* (2002), Kojekin and Savchenko (2002), Kozhekin *et al.* (2003) presented algorithms for image retouching. The corrupted image is restored by using the space-mapping technique. Their algorithm is implemented in three-dimensional space. The Cholesky decomposition is used for solving linear equations. The input image is handled as three separated colour channels (R, G, B). Wang and Qin (2006) improved their approach. Firstly, the two-dimensional image is converted to a three-dimensional cloud of points. Then, an implicit surface is reconstructed from these points. Finally, the Radial-basis function is applied to reconstruct the surface, and consequently, the damaged parts of the image are restored. Uhlíř and Skala (2006) also applied this approach. In their algorithm, the known pixels in the squared-window of constant size are used to construct a system of linear equations.

3. Radial-Basis Function Interpolation Using the Processing-Circle

Our method follows the approaches based on Radial-basis function interpolation (Morse *et al.*, 2001; Savchenko *et al.*, 2002; Kojekin and Savchenko, 2002; Kozhekin *et al.*, 2003; Wang and Qin, 2006; Uhlíř and Skala, 2006). These types of interpolation are important techniques for data interpolation and approximation (Buhmann, 2003). Therefore, the Radial-basis function could be used for interpolating a function f with n points and by using n radial basis functions centered at these points. Wendland constructed compact, locally-supported radial basis functions (CSRBF), which guarantee that the system of linear equations (1) is positive-definite. Therefore, a solution for the linear system always exists (Wethland, 1995; Morse *et al.*, 2001). Compactly-supported radial basis functions also reduce computational complexity.

In continuation, Radial-basis function interpolation is introduced together with its use in the reconstruction of a corrupted pixel. Adopting Kojekin and Savchenko (2002),

the following interpolation schema is needed:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \phi\big(|\mathbf{x} - \mathbf{c}_i|\big) + ax + by + c, \tag{1}$$

$$\sum_{1}^{n} \lambda_i c^x = \sum_{1}^{n} \lambda_i c^y = \sum_{1}^{n} \lambda_i = 0, \tag{2}$$

where $[\lambda_1 \ldots, \lambda_n,\ a,\ b,\ c]$ is the solution of linear system of equations. λ_i represents the weight of the radial basis function positioned at point c_i. Parameters a, b, c represent the constant portion of f, and ensure a positive definiteness of the solution (Morse *et al.*, 2001). Equations (2) ensure the orthogonality of a solution (Uhlíř and Skala, 2006). $\mathbf{x} = (x, y)$ are the coordinates of the corrupted pixel, ϕ are the Radial-basis functions, and n is the number of known pixels within the neighbourhood of the corrupted pixel. It also represents the number of Radial basis functions. The values of the basis functions can be calculated in advance. The values of function f for input coordinates $\mathbf{c}_i$ are known pixel intensity values h_i. In order to calculate intensity value of the corrupted pixel, we have to find the solution for the following linear system of equations, as obtained from (1) and (2):

$$\begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \ldots & \phi_{1,n} & c_1^x & c_1^y & 1 \\ \phi_{2,1} & \phi_{2,2} & \ldots & \phi_{2,n} & c_2^x & c_2^y & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{n,1} & \phi_{n,2} & \ldots & \phi_{n,n} & c_n^x & c_n^y & 1 \\ c_1^x & c_2^x & \ldots & c_n^x & 0 & 0 & 0 \\ c_1^y & c_2^y & \ldots & c_n^y & 0 & 0 & 0 \\ 1 & 1 & \ldots & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ a \\ b \\ c \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3}$$

where $\phi_{i,j} = \phi(|\mathbf{c}_i - \mathbf{c}_j|),\ i, j = 1, \ldots, n$ and h_i are the pixel intensity values at the coordinates $\mathbf{c}_i$. The solution of linear system of equations is $[\lambda_1, \lambda_2, \ldots, \lambda_n, a, b,$ and $c]$. These parameters are used to calculate the function $f(\mathbf{x})$, the values of which represent the value of the corrupted pixel (in the case of the colour image, the calculation is done separately for each component).

3.1. *Corrupted Pixel Reconstruction with a Processing-Circle*

In our approach, the image is reconstructed from the middle of the image towards its borders. Previous approaches have used a processing-square to select the known pixels that surround the corrupted pixel. The processing-circle is applied in our approach. In Fig. 1 the corrupted pixels are plotted in grey. The unknown pixel that is going to be reconstructed, is in the centre of the circle and marked by $\times$. When the reconstructed pixel becomes known, it is used to reconstruct the next corrupted pixels. During the processing-circle approach, the distance d is used to determine whether the known pixel c_i is inside

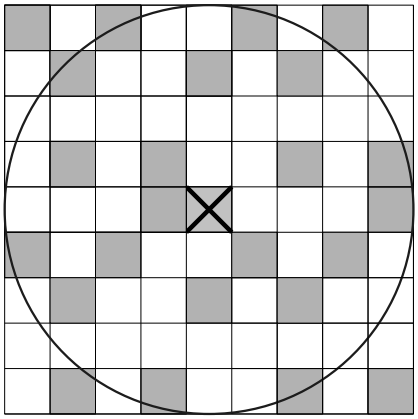


Fig. 1. The processing-circle inside the processing-square.

the circle with radius r and centre $\mathbf{x}$:

$$d = |\mathbf{c}_i - \mathbf{x}|. \tag{4}$$

If the distance d is smaller than radius r, the pixel at coordinate c_i participates in the process of determining the value of the corrupted pixel, i.e., it is used for constructing a system of linear equations (3). The size and the computational time of this system of linear equations depends on the number of input values (known pixel values within the neighbourhood of the corrupted pixel). What this means in practise is given in Section 4.

3.2. *Data-Dependent Pixel Corruption*

Most photographic images contain areas with pixels that have the same or almost the same values. This feature can be exploited to corrupt pixels in a data-dependent way, i.e., more pixels are corrupted in these areas where more pixels have similar values. The following approach is proposed for this. A squared-window of size w is formed (Fig. 2) and the mean pixel value within this square is determined. Next, the differences between each pixel's value and the pixels' mean value within the squared-window are calculated. Firstly, those pixels that have smaller differences than the predefined threshold th are identified. Then, the pixel with the smallest differences is found from among them and is marked as non-corrupted, and the others as the opposite. Secondly, the pixel that has a larger difference than th is marked as non-corrupted. The square is then moved for distance m. If $m <= w$, the new square position could partially overlap with the previ ous square position. For example, the grey square in Fig. 2 overlaps the black square. The status of the pixel is then reconsidered within overlapping region. As a result, more pixels from those areas where pixel intensity values change slowly are corrupted, and oppositely, more pixels are left unchanged in those areas where the pixel values are diverse.

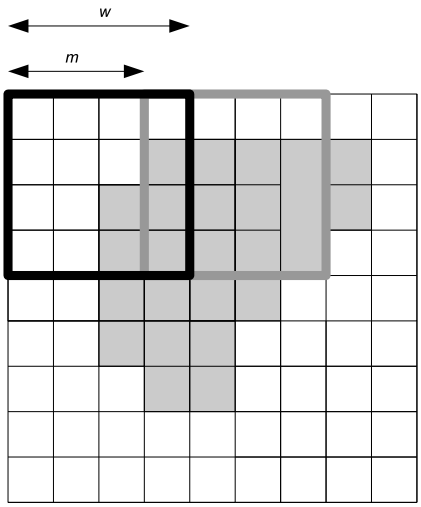


Fig. 2. Data-dependent corrupting of pixels ($w = 4$).

4. Results

Several experiments have were carried out to evaluate the proposed method.

1. The influence of various Radial-basis functions was firstly evaluated. An artificial and three standard photorealistic images were used for the tests (Fig. 3). 50% of the image pixels were randomly damaged and then reconstructed using the processing-circle approach. A processing-circle of size 5 was used during this experiment. This circle-size was used because a symmetrical area was needed around the corrupted pixel. Therefore, the circle-size need to be odd. A smaller circle-size, for example 3, would have been insufficient because the number of known pixels within the neighbourhood of the corrupted pixel would have been too small for its correct reconstruction. In contrast, a higher circle-size, for example 7, would have increased the number of known pixel within the neighbourhood of the corrupted pixel. Therefore, the the computational time increased (as shown in Table 2). In order to compare the results, the same metric was used a s in the article (Uhlíř, 2006). The Mean Absolute Error (MAE) was used, i.e., the city-block metric:

$$\delta = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left| \Omega_1(i,j) - \Omega_2(i,j) \right|. \tag{5}$$

Table 1 shows reconstruction times and the differences between the Ω_1 and Ω_2 for different Radial-basis functions and standard testing images. The input parameter r was the normalised distance from the centre of the processing-circle. The experiments were executed on a computer with an Intel Core2 2.66 GHz processor. The results in Table 1 show that the different Radial-basis functions did not have much influence on the reconstruction time and the quality of image reconstruction.

Table 1
The results of reconstruction with different Radial-basis functions $\phi(r)$

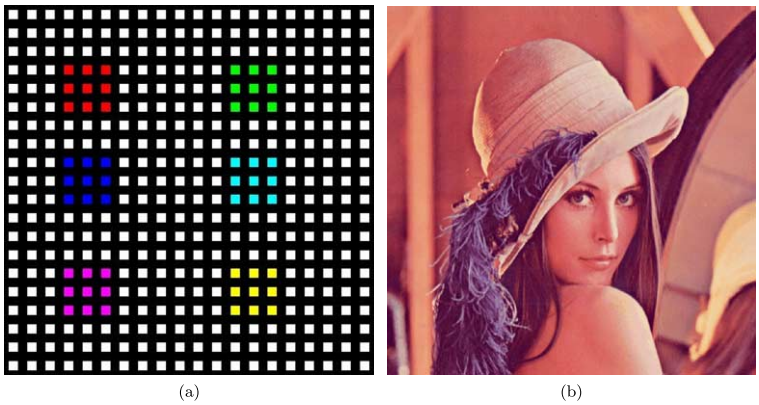
$\phi(r)$	Lenna		Baboon		Pepper	
	$t(s)$	δ	$t(s)$	δ	$t(s)$	δ
$r^2 \log r$	1.7350	2.5240	1.7340	8.4167	2.4060	4.2491
$(1-r)^3$	1.6720	2.5410	1.7030	8.2863	2.4060	4.1020
$1-r$	1.6720	2.5342	1.8600	8.2843	2.4060	4.4589
$\sqrt{r^2+10}$	1.734	3.4615	1.8120	10.7599	2.3910	6.8465
r^3	1.718	2.6332	1.7650	8.7929	2.4060	4.4589
$\exp(-r^2)$	1.703	3.0789	1.7650	9.9363	2.3910	5.2985
$(1-r)^2$	1.656	2.5423	1.7500	8.2840	2.3900	4.1090

Table 2
The results of reconstruction using image from Fig. 4a

Noise ratio	Size	Method	$t(s)$	δ	Δt (%)	$\Delta\delta$ (%)
0.1	5	Square	0.640	1.006	36.6	98.3
		Circle	0.406	0.989		
	7	Square	2.266	1.180	44.8	101.2
		Circle	1.250	1.194		
0.5	5	Square	1.625	7.530	34.6	100.1
		Circle	1.063	7.538		
	7	Square	5.219	7.675	43.7	101.0
		Circle	2.938	7.753		
0.9	5	Square	2.563	30.441	25.6	101.1
		Circle	1.906	30.763		
	7	Square	6.391	27.572	42.1	102.6
		Circle	3.703	28.285		

Therefore, it was decided to use the Radial-basis function $r^2 \log r$ during the continuation. In the next experiments tested the influence of processing-circle against the processing-square.

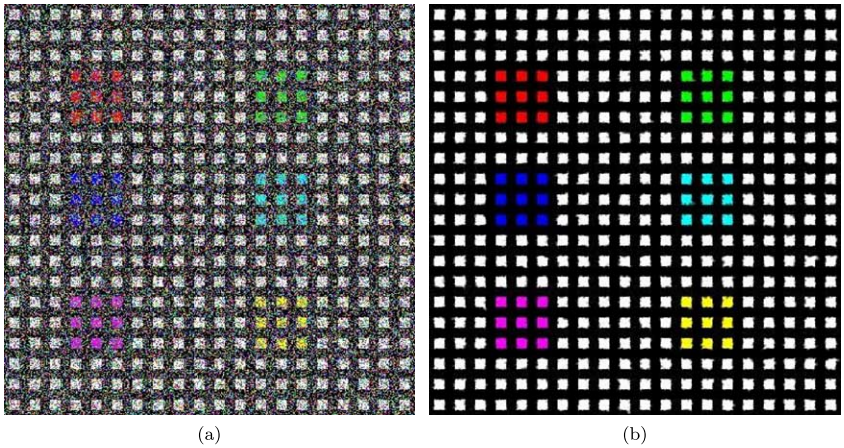
2. The method was also tested on an artificial image, as shown in Fig. 3a. The image size was 400×400 pixels. Figure 4b shows the reconstructed image obtained from the image with 50% of corrupted pixels (Fig. 4b). Table 2 presents the numerical results. It also shows the reconstruction times for various noise ratios, the processing-square or circle-sizes (t_w and t_c are the reconstruction times for the square and the circle, respectively). The percentage of time reduction $\Delta t = 100(t_w - t_c)/t_w$ was calculated, too. Similarly, the differences δ_c and δ_w



(a) (b)

(c) (d)

Fig. 3. (a) Artificial image; (b) Lenna; (c) Baboon; (d) Pepper.



(a) (b)

Fig. 4. Reconstruction of an artificially created image: (a) image with 50% randomly corrupted pixels; (b) the results of reconstruction.

Table 3
The results of reconstruction from corrupted image Fig. 5a

Noise ratio	Size	Method	$t(s)$	δ	Δt (%)	$\Delta\delta$ (%)
0.1	5	Square	0.656	0.404	33.4	101.2
		Circle	0.437	0.408		
	7	Square	0.437	0.405	45.4	100
		Circle	1.219	0.405		
0.5	5	Square	2.516	2.509	34.1	101.2
		Circle	1.657	2.538		
	7	Square	8.000	2.445	44.0	99.8
		Circle	4.484	2.441		
0.9	5	Square	3.734	9.486	27.2	110.5
		Circle	2.719	10.483		
	7	Square	9.765	7.826	42.1	105.5
		Circle	5.656	8.253		

between the original and reconstructed-images is given, together with the calculated percentage $\Delta\delta = 100(\delta_c/\delta_w)$.

3. The image of Lenna (Fig. 3b) was used in the next experiment. The reconstructed image is presented in Fig. 5b. It was obtained from an image with 64.73% of corrupted pixel (Fig. 5a). Table 3 shows the numerical results of the reconstruction for the same parameter values as in the previous experiment.

4. The effect of the data-dependent corruption of pixels was observed using several different threshold values th, square sizes w and square moving distances m. The threshold value was the same for each processing-circle/square and was not recalculated during the reconstruction process. The image was reconstructed by using processing-circle of size 5. The image shown in Fig. 5c was obtained after corrupting the original image (Fig. 3b) using the data-dependent approach. Figure 5d presents the reconstructed image. Table 4 shows the numerical results, where the data-dependent pixel corruption with different parameter values and the tested-image of Lenna were applied. The value n_c represents the number of corrupted pixels within the tested image. The results for other standard testing images are showed in Table 5. The reconstruc ted images are presented in Figs. 6 and 7. The results were compared with the random pixel corruption when using the same number of corrupted pixels n_c and the reconstruction time as t. The differences between the reconstruction times were calculated as $\Delta t = 100(t_u - t_d)/t_u$, where t_u was the reconstruction time for random pixel corruption and t_d the time for the data-dependent corrupted pixels. Similarly, the ratio $\Delta\delta$ was calculated as $\Delta\delta = 100(\delta_u/\delta_d)$, where δ_u was the difference between the original and reconstructed images for random pixel corruption, and δ_d was the same difference for the data-dependent pixel corruption.

Table 4
The results of reconstruction of Lenna, if the data-dependent pixel removal is used

Method	th	n_c	n_c (%)	$t(s)$	Δt (%)	δ	$\Delta\delta$ (%)
$w = 5, m = 3$							
Data	0.5	7222	2.94	0.219	−27.33	0.0492	237.44
Random	–			0.1720		0.1167	
Data	5	92,690	37.72	1.3910	− 8.59	1.2811	137.23
Random	–			1.2810		1.7581	
Data	20	159,069	64.73	2.0480	− 0.79	3.1853	116.83
Random	–			2.0320		3.7214	
$w = 7, m = 3$							
Data	0.5	6687	2.72	0.1560	0.00	0.0501	211.65
Random	–			0.1560		0.1061	
Data	5	83,517	33.98	1.2030	0.00	1.2201	128.11
Random	–			1.2030		1.5630	
Data	20	154,517	62.87	1.8440	3.30	3.1694	111.89
Random	–			1.9070		3.5462	
$w = 4, m = 2$							
Data	0.5	3983	1.62	0.1250	−14.68	0.0244	265.5
Random	–			0.1090		0.0649	
Data	5	68,674	27.94	1.0620	− 4.53	0.8059	154.55
Random	–			1.0160		1.2456	
Data	20	119,438	48.60	1.5630	5.62	2.1945	110.91
Random	–			1.6560		2.4340	

5. Discussion

The results from the first experiment show that the different Radial-basis functions have very little influence on the reconstruction time and the quality of image reconstruction. Therefore, the function $\phi(r) = r^2 \log r$ was applied during the rest of the experiments, as used elsewhere (Savchenko *et al.*, 2002; Kojekin and Savchenko, 2002; Uhlíř and Skala, 2006).

The results listed in Tables 2 and 3 show that when the processing-circle was used, the reconstruction time was considerably reduced (from 25.6% up to 44% and from 27.2% up to 45.4%, respectively). This percentage depended on the square-size and the noise-ratio. Both methods needed more time when the noise ratio was higher. But, the processing-

Table 5
The results of reconstruction for various testing images

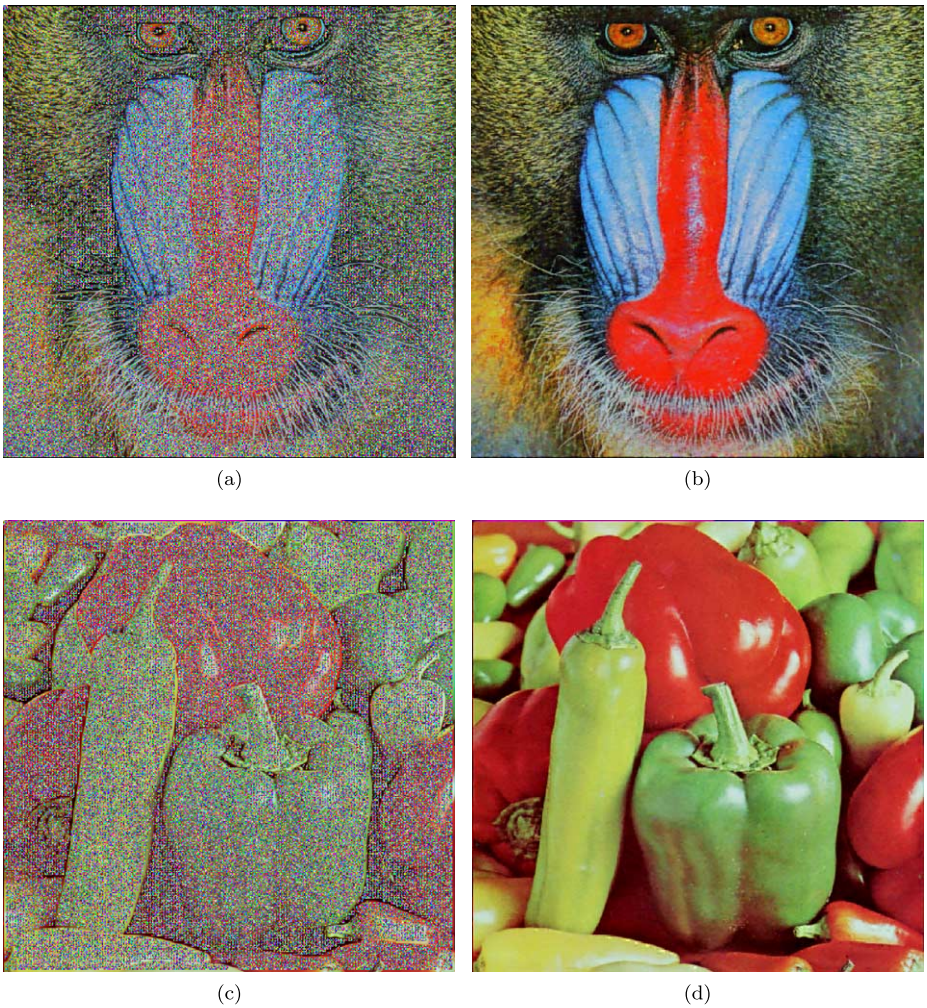
Image	Method	n_c	$n_c(\%)$	$t(s)$	$\Delta t\ (\%)$	δ	$\Delta\delta\ (\%)$
$th = 20,\ w = 5, m = 3$							
Lenna	Data	$159,069$	64.73	2.048	-0.78	3.1853	116.83
	Random			2.032		3.7214	
Baboon	Data	$118,940$	49.55	1.8280	3.39	6.3916	130.45
	Random			1.8900		8.3379	
Pepper	Data	$173,673$	66.25	2.7180	-10.89	3.6517	115.41
	Random			2.4220		4.2143	
$th = 10,\ w = 5, m = 2$							
Lenna	Data	$98,827$	40.21	1.6720	-6.58	1.5411	123.62
	Random			1.5620		1.9051	
Baboon	Data	$54,315$	22.63	1.0630	-10.35	2.2595	155.80
	Random			0.9530		3.5203	
Pepper	Data	$114,422$	43.65	1.8280	-3.39	2.1859	116.73
	Random			1.7660		2.5515	

circle was relatively less demanding. The experiments showed that the image itself had a negligible effect on the computational time. It could be concluded, that regardless of the image, the processing time is always significantly lower than if the processing-square is used without spoiling the quality of image reconstruction.

Tables 4 and 5 present the results of the image reconstruction when using the data-dependent pixel corruption. After the reconstruction, the difference between the original and the reconstructed image was smaller than when using the random pixel corruption. The lower threshold values caused the reconstruction time to increase, but the difference between the original and the reconstructed images was smaller. In contrast, the reconstruction time decreased with higher threshold values and the difference was greatest between the original and reconstructed images.

Table 6 shows the results of the comparison between the data-dependent pixel corruption approach using the processing-circle and the random pixel corruption using the processing-square. It can be seen that the reconstruction time of our method was shorter. Similarly, the differences between the original and reconstructed images were smaller when using our approach. Therefore, a combination of the processing-circle approach and data-dependent corruption reduces both – the reconstruction time and the difference between an original and a reconstructed image.

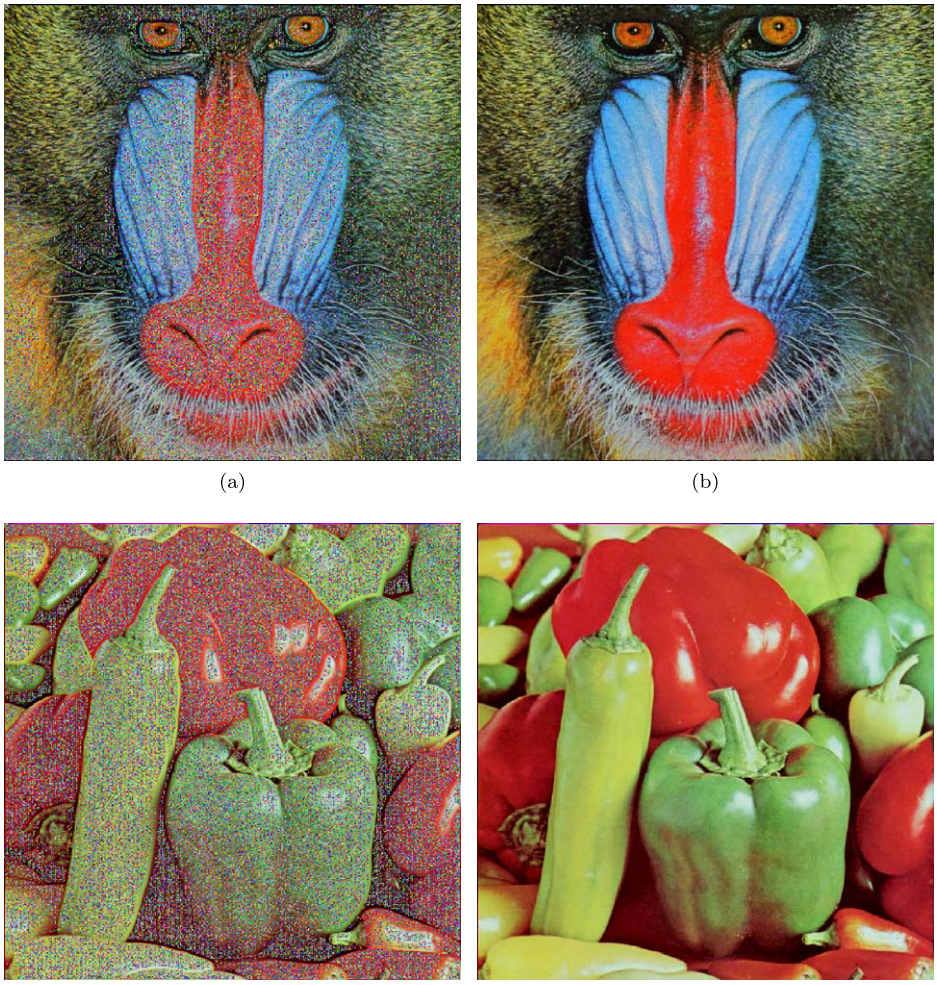
Fig. 5. (a) The photograph of Lenna with 64.73% randomly corrupted pixels; (b) reconstructed image from corrupted image (a); (c) the corrupted Lenna with 64.73% data-dependently corrupted pixels; (d) the results of reconstruction, if the corrupted image in (c) is used.



(a) (b)

(c) (d)

Fig. 6. Reconstruction of data-dependently corrupted pixel ($th = 20,\ w = 5,\ m = 3$): (a) the photograph of Baboon with 49.55% data-dependently corrupted pixels; (b) reconstructed image from corrupted image (a); (c) the photograph of Pepper with 66.25% data-dependently corrupted pixels; (d) the results of reconstruction, if the corrupted image in (c) is used.



(a) (b)

(c) (d)

Fig. 7. Reconstruction of data-dependently corrupted pixel ($th = 10$, $w = 5$, $m = 2$): (a) the photograph of Baboon with 22.63% data-dependently corrupted pixels; (b) reconstructed image from corrupted image (a); (c) the photograph of Pepper with 43.65% data-dependently corrupted pixels; (d) the results of reconstruction, if the corrupted image in (c) is used.

Table 6

The comparison between the data-dependent pixel corruption approach using the processing-circle and the random pixel corruption using the processing-square

Method	th	n_c	n_c (%)	$t(s)$	Δt (%)	δ	$\Delta\delta$ (%)
$w = 5, m = 3$							
Data/circle	2	38,833	15.80	0.75	6.25	0.3577	184.79
Random/square	–			0.80		0.6610	
Data/circle	15	151,058	61.47	1.38	38.9	2.8666	116.52
Random/square	–			2.26		3.3402	
Data/circle	30	167,317	68.08	1.66	33.3	3.6421	109.13
Random/square	–			2.49		3.9747	

6. Conclusion

An efficient modification of a method for reconstructing raster images with deliberately corrupted pixels is presented. Instead of the processing-square, we have proposed the processing-circle. The reconstruction time has been significantly reduced without affecting the quality of the reconstructed image. In continuation, a simple data-dependent approach for pixel corruption is also proposed. Those areas within the image that have similar pixel values have been corrupted more intensively. We have shown that a combination of the processing-circle and data-dependent pixel corruption gives superior results regarding the time needed for the reconstruction and the quality of the reconstructed image.

References

Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C. (2000). Image inpainting. In: Akeley, K. (Ed.), *Siggraph 2000*, *Computer Graphics Proceedings, Annual Conference Series*. ACM Press/ACM SIGGRAPH, Addison/Wesley/Longman, New Orleans, pp. 417–424.

Bornard, R., Lecan, E., Laborelli, L., Chenot, J.H. (2002). Missing data correction in still images and image sequences. In: *Proceedings of the Tenth ACM International Conference on Multimedia (MM-02)*. ACM Press, New York, pp. 355–361.

Bornemann, F., März, T., (2007). Fast image inpainting based on coherence transport. *Journal of Mathematical Imaging and Vision*, 28(3), 259–278.

Buhmann, M.D. (2003). *Radial Basis Functions*. Cambridge University Press, Cambridge.

Chen, N., Reiter, C.A. (2007b). Best analogs for replacing missing image data. *Computers & Graphics*, 31(4), 617–624.

Chen, Q., Zhang, Y.X., Liu, Y.C. (2007a). Image inpainting with improved exemplar-based approach. In: Sebe, N., Liu, Y., Zhuang, Y., Huang, T.S. (Eds.), *Multimedia Content Analysis and Mining*, *Lecture Notes in Computer Science*, Vol. 4577. Springer/Weikai, Berlin/China, pp. 242–251.

Criminisi, A., Pérez, P., Toyama, K. (2004). Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9), 1200–1212.

Demanet, L., Song, B., Chan, T. (2003). *Image Inpainting by Correspondence Maps: A Deterministic Approach*. Tech. rep., California Institute of Technology.

Efros, A.A., Leung, T.K. (1999). Texture synthesis by non-parametric sampling. In: *The Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. IEEE Computer Society, Kerkyra, pp. 1033–1038.

Kojekin, N., Savchenko, V. (2002). Using CSRBFs for surface retouching. In: Villanueva, J. (Ed.), *Proceedings of the 2nd IASTED International Conference on Visualization, Imaging and Image Processing*, Marbella, Spain. Benalmadena, Malaga, pp. 613–618.

Kozhekin, N., Savchenko, V.V., Senin, M., Hagiwara, I. (2003) An approach to surface retouching and mesh smoothing. *The Visual Computer*, 19(7–8), 549–564.

Morse, B.S., Yoo, T.S., Rheingans, P., Chen, D.T., Subramanian, K.R. (2001). Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. *Shape Modeling International*, 89–98.

Oliveira, M.M., Bowen, B., McKenna, R., Chang, Y.S. (2001). Fast digital image inpainting. In: Hamza, M.H. (Ed.). *Proceedings of the IASTED International Conference on Visualization, Imaging and Image Processing (VIIP 2001)*. ACTA Press, Marbella, pp. 261–266.

Savchenko, V.V., Kojekine, N., Unno, H. (2002). A practical image retouching method. In: *Proceedings of the First International Symposium on Cyber Worlds*. IEEE Computer Society, Los Alamitos, USA, pp. 480–487.

Shih, T.K., Chang, R.C., Lu, L.C., Ko, W.C., Wang, C.C. (2004). Adaptive digital image inpainting. In: Barolli, L. (Ed.), *18th International Conference on Advanced Information Networking and Applications*, Vol. 1. IEEE Computer Society, Fukuoka, pp. 71–77.

Sprott, J.C. (2004). A method for approximating missing data in spatial patterns. *Computers & Graphics*, 28(1), 113–117.

Telea, A. (2004). An image inpainting technique based on the fast marching method. *Journal of Graphics Tool*, 9(1), 23–34.

Uhlíř, K., Skala, V. (2006). Radial basis function use for the restoration of damaged images. In: Wojciechowski, K., Smolka, B., Palus, H., Kozera, R., Skarbek, W., Noakes, L. (Eds.), *Computer Vision and Graphics International Conference*, Vol. 32. Springer, Berlin, pp. 839–844.

Wang, W., Qin, X. (2006). An image inpainting algorithm based on CSRBF interpolation. *International Journal of Information Technology*, 12(6), 112–119.

Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree *Advances in Computational Mathematics*, 4(1), 389-396.

Wu, J., Ruan, Q., An, G. (2010) Exemplar-based image completion model employing PDE corrections. *Informatica*, 21(2), 259–276.

Yamauchi, H., Haber, J., Seidel, H.P. (2003). Image restoration using multiresolution texture synthesis and image inpainting. In: *Computer Graphics International*. IEEE Computer Society, Los Alamitos, pp. 120–125.

B. Lipuš is a teaching assistant at the Faculty of Electrical Engineering and Computer Sciences, University of Maribor. He received his PhD in 2005. He worked as a software developer in industry for two and half years. He is a member of the Laboratory for Geometric Modelling and Multimedia Algorithms. His research interests include computer graphics, computer-aided geometric design, image reconstruction and data compression.

B. Žalik is a full professor of computer science at the Faculty of Electrical Engineering and Computer Science, University of Maribor and the head of the Laboratory for Geometric Modelling and Multimedia Algorithms. He is currently the vice-dean for research at the faculty and a member of management board of the Slovenian Research Agency. He is the author or co-author of 60 journal papers and more than 100 conference papers. His research interests include data visualization, geometric modelling, computational geometry, multimedia, data compression, and GIS.

Efektyvi skaitmeninių vaizdų rekonstrukcija esant tyčiniam vaizdo taškų sugadinimui

Bogdan LIPUŠ, Borut ŽALIK

Šiame straipsnyje pristatomas naujas metodas, gebantis rekonstruoti tyčia sugandintus skaitmeninio rastrinio vaizdo taškus. Pristatomas metodas yra greitesnis lyginant su kitais, kadangi, skirtingai nei kiti metodai, šis naudoja skritulio, o ne kvadrato formos rekonstrukcinį elementą. Autoriai straipsnyje atskleidžia, kad pakeistus rekonstrukcinio elemento kvadrato formą į skritulio rekonstrukcijos kokybė dėl to nenukenčia. Vaizdo rekonstrukcijos kokybė papildomai gerinama analizuojant pradinio vaizdo taškų sugadinimo pobūdį. Straipsnyje parodyta, kad skritulio formos rekonstrukcinis elementas ir žinios apie taškų sugadinimą leidžia pagreitinti vaizdo rekonstrukcijos procesą bei sumažina klaidingai atstatytų skaitmeninio vaizdo taškų kiekį.