DE GRUYTER DOI 10.1515/jisys-2013-0031 — Journal of Intelligent Systems 2013; 22(3): 335–350

Research Article

Rajesh Pandurang Borole* and Sanjiv Vedu Bonde

Patch-Based Inpainting for Object Removal and Region Filling in Images

Abstract: A large number of articles have been devoted to the application of "texture synthesis" for large regions and "inpainting" algorithms for small cracks in an image. A new approach that allows the simultaneous filling in of different structures and textures is discussed in this present study. The combination of structure inpainting and patch-based texture synthesis carried out (termed as "patch-based inpainting") for filling and updating the target region shows additional advantages over earlier approaches. The algorithm discussed here uses the patch-based inpainting with isophote-driven patch-based texture synthesis at the core. In this algorithm, once the user selects the regions to be restored, the algorithm automatically searches and fills in these regions with the best matching information surrounding them. We have assigned high priorities to the pixels on the boundary and the structure by computing data terms $D(p)$, and the texture and corners are prioritized by computing the confidence $C(p)$ of the pixel. We also regularized and weighted the confidence of the pixels, $RC(p)$, to achieve a balance of the two. The patch search area near the pixel patch to be filled is bounded for algorithm speed improvement. Patch-based filling significantly improve execution speed compared with pixel-based filling. Filling in is done in such a way that the structure information arriving at the region boundaries is propagated inside. A number of examples on real and synthetic images are used to demonstrate the effectiveness of the algorithm. Robustness with respect to the shape of the selected target region is also demonstrated.

Keywords: Object removal, region filling, patch-based inpainting, simultaneous texture, structure propagation.

***Corresponding author: Rajesh Pandurang Borole,** Department of Electronics and Telecommunication Engineering, Shri Guru Gobind Singhji Institute of Engineering and Technology, Vishnupuri, Nanded, Maharashtra 431 606, India, e-mail: rpborole@yahoo.com
Sanjiv Vedu Bonde: Department of Electronics and Telecommunication Engineering, Shri Guru Gobind Singhji Institute of Engineering and Technology, Vishnupuri, Nanded, Maharashtra 431 606, India

1 Introduction

The two most commonly used graphical techniques to fill the gap after object removal are image inpainting and texture synthesis. Digital image inpainting refers to the inpainting process performed on digitized images. It is important to have an algorithm that can automatically fill in a region marked by the user. The user provides the image with a marked region to be inpainted, which is referred to as the mask, in a specified color. The filling in of these marked regions by the computer algorithm should result in an image that looks natural and undistorted. Object removal from images is an image manipulation technique. Removing objects from images starts with the masking out of the undesired object and by making a gap at the area occupied by that object. Then this gap is filled using digital inpainting techniques.

Texture replication and propagation are used to fill in gaps in images. Different approaches were developed for synthesizing textures, including statistical and image-based methods. The application of these methods for filling the gaps in an image is called constrained texture synthesis, and it is used for filling images having more textured areas. Neither structural inpainting nor texture synthesis alone can provide the ultimate solution, but the combination of these two produces good results. Image inpainting considers the image as a collection of structures, shapes, and objects that are separated from one another by sharp edges, but each one is itself smooth, and texture considers the low-stochasticity features of the image. Various approaches are introduced to reconstruct images using inpainting techniques, such as level lines, PDE-based inpainting, fluid interpolation, Euler's elastic, bounded variation, and heat transfer [2, 3, 4, 7, 19].

All such methods have their own advantages and limitations. The proposed patch-based inpainting approach overcomes many of the limitations of earlier approaches.

Automatic digital image inpainting first brought into the field of image processing by Bertalmio et al. [4]. This algorithm imitates the traditional inpainting processes, such as identification of the area to be corrected, marking of the boundary of a region to be filled, and continuing the lines of similar color. These techniques are effective only for small scratches. The algorithms work well in images that are relatively smooth and have low noise or texture. The process cannot fill in regions that are highly textured or that contain large noise, which leads to noncontinuation and blurring of the edges. Since then, there are a number of algorithms proposed to solve the inpainting problem [2, 7, 18]. These are used to fill only narrow gaps in images. They fail to reconstruct large damaged regions in the images. Motion estimation and autoregressive models to interpolate losses in films from adjacent frames are used [16]. The basic idea is to copy the right pixels from neighboring frames into the gap. The technique cannot be applied to still images or films, where the regions to be inpainted span over many frames.

The work [5] decomposes the original image into two components, which are processed separately by inpainting and texture synthesis. The outputs of the two are summed up as the resultant image. This approach remains limited to the removal of small image cracks, and the diffusion process continues to blur the filled region. Drori et al. [10] describes an algorithm that interleaves a smooth approximation with example-based detail synthesis for image completion. Both of these proposed algorithms were extremely slow. The first attempt to use exemplar-based synthesis specifically for object removal was proposed [14]. They proposed that the order in which a pixel in the target region is filled is dictated by the level of "texturedness" of the pixel's neighborhood. Although the intuition is sound, strong linear structures were often overruled by nearby noise, thus reducing the value of the extra computation. A related technique drove the fill order by the local shape of the target region but fails to explicitly propagate linear structures [6].

Several researchers have considered texture synthesis as a way to fill large image regions with "pure" textures. Pure textures are defined as repetitive two-dimensional textural patterns with moderate stochasticity. This is based on a large research carried out on texture synthesis, which seeks to replicate the texture from a given small source sample of pure texture [9, 13, 15]. These exemplar-based techniques cheaply and effectively generate new texture by sampling and copying color values from the source [1, 12, 17]. It only contains the process to replicate textured areas in the images. In case of regular textures, exemplar-based inpainting works well but fails to replicate structures in the still images.

In contrast with previous approaches, there is a need for a technique that does not require the user to specify where the information to fill comes from and instead be automatically be accomplished by the algorithm. The technique should allow to simultaneously fill in numerous regions containing completely different structures and surrounding backgrounds. In addition, no limitations should be imposed on the topology of the region to be inpainted. Therefore, the patch-based approach of Criminisi et al. [8] seems more appropriate, and a further modification is being proposed in the present article. This results in an algorithm that has the efficiency and qualitative performance of exemplar-based texture synthesis as well as linear structure inpainting.

2 Key Observations and Algorithm

Both image inpainting and texture synthesis have their strengths and weaknesses. Image inpainting, which extends the linear structure into the gap using the isophote information of the boundary pixels, leads to a natural propagation of the linear structures into the gap. However, because the extension is actually using

diffusion techniques, artifacts (e.g., blur) could be introduced. Meanwhile, texture synthesis copies the pixels from the existing parts of the image, thus avoiding blur. The shortcoming of texture synthesis is that it focuses on the whole image space and not on resolving the priorities of pixels on texture and linear structures around the boundary of the gap. As a result, texture propagation dominates, and linear structures are not naturally extended into the gap. The result would likely have distorted lines and noticeable differences between the gap and its surrounding area.

One interesting observation is that although image inpainting and texture synthesis seem to differ, they might actually complement each other. If we could combine the advantages of both approaches, we would get a clear gap filling that is the natural extension from the surrounding area. Criminisi et al. [8] proposed an algorithm that exactly does this. They used the sampling concept of Efros et al. [11] and improved on it by considering the isophotes, giving a higher priority to those points that are on the boundary of the gap as well as on structures, isophotes, and corners, which leads to their natural extension into the gap. To identify those points, Criminisi et al. [8] gives a priority value to all the pixels on the boundary of the gap. These important points on the structure and corners will get higher priorities according to the algorithm, and thus, the linear structures would be extended first. For each pixel on the boundary, a patch is constructed with that pixel at the center (Figure 1). The patch's priority is the product of two elements: a confidence term $C(p)$ and a data term $D(p)$.

2.1 Computing Priorities of the Patch

Our filling algorithm assigns high priorities to the pixels on corners via the confidence term and simultaneously assigns priorities to pixels on isophotes via data terms; thus, the highest-priority-pixel patch is filled first.

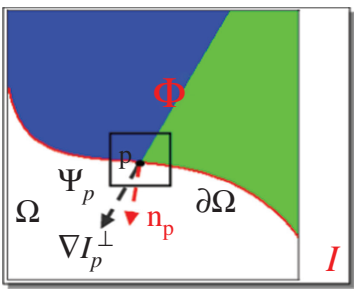

Figure 1. Notation Diagram.
Given Patch ψ_p,n_p is Normal to the Contour $\partial\Omega$ of the Target Region Ω and $I_p^{\perp}$ is the Isophote (Direction and Intensity) at Point *P*. The Source Region is Denoted by ϕ. The Entire Image is Denoted with *I*.

$$C(p) = \frac{\sum_{q \in \psi p \cap \phi} C(q)}{|\psi_p|}, \quad 0 \le C(p) \le 1 \tag{1}$$

$$D(p) = \frac{\nabla I_p^{\perp}}{\alpha}, \quad 0 \le D(p) \le 1, \tag{2}$$

where α is the normalization factor (e.g., $\alpha = 255$).

Priorities are calculated as

$$P(p) = C(p) \times D(p), \tag{3}$$

where $C(p)$ describes the number of pixels already filled in the patch. Figure 2 shows that the patches, including the corners and thin tendrils, of the target region have a higher priority and are filled first.

$D(p)$ describes how strong the isophote is in hitting the boundary. This term boosts the priority of a patch that an isophote "flows" into. $D(p)$ is especially important because it encourages linear structures to be synthesized first and are thus propagated securely into the target region.

The confidence of the source region pixels is initially set to 1 and that of the gap region is set to 0. Once the boundary pixel patch ψ_p is filled, the newly filled pixels of the target region of ψ_p will be assigned the same confidence of p. As the algorithm propagates in target region, the confidence value decreases exponentially, which makes the computed priorities indistinguishable, resulting in an incorrect filling order. Even if the data values are increasing, the shape of the priority curve is dominated by exponentially decreasing confidence values. This effect is more noticeable when filling a large target region. This intense dominance is due to the effect of multiplication. Thus, the original priority in the multiplicative form needs to be changed to avoid a pronounced decreasing effect. The

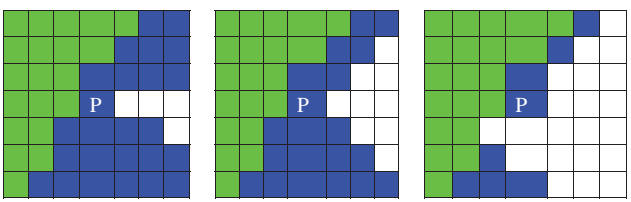

Figure 2. Confidence of Pixel p with a 7×7 Patch.
White Indicates the Pixels of the Target Region. Pixels on Sharp Corners have High Confidence and hence are Filled on Priority, which Leads to a Simultaneous Neighboring Structure and Texture Propagation in the Gap.

additive form, Eq. (4), with weight constants is more linearly proportional and stable to unexpected changes.

The additive form of the fill priorities is

$$P(p)=C(p)+D(p). \tag{4}$$

The direct combination of $C(p)$ and $D(p)$ is unreasonable because of the significant difference in their values. The priority values are dominated by confidence values as compared to values. As the algorithm propagates, the rate of $C(p)$ significantly decreases (Figure 3). Therefore, it is regularized and weighted:

$$RC(p)=(1-\omega)\times C(p)+\omega, \quad 0\le\omega\le 1, \tag{5}$$

where ω is a regularizing factor.

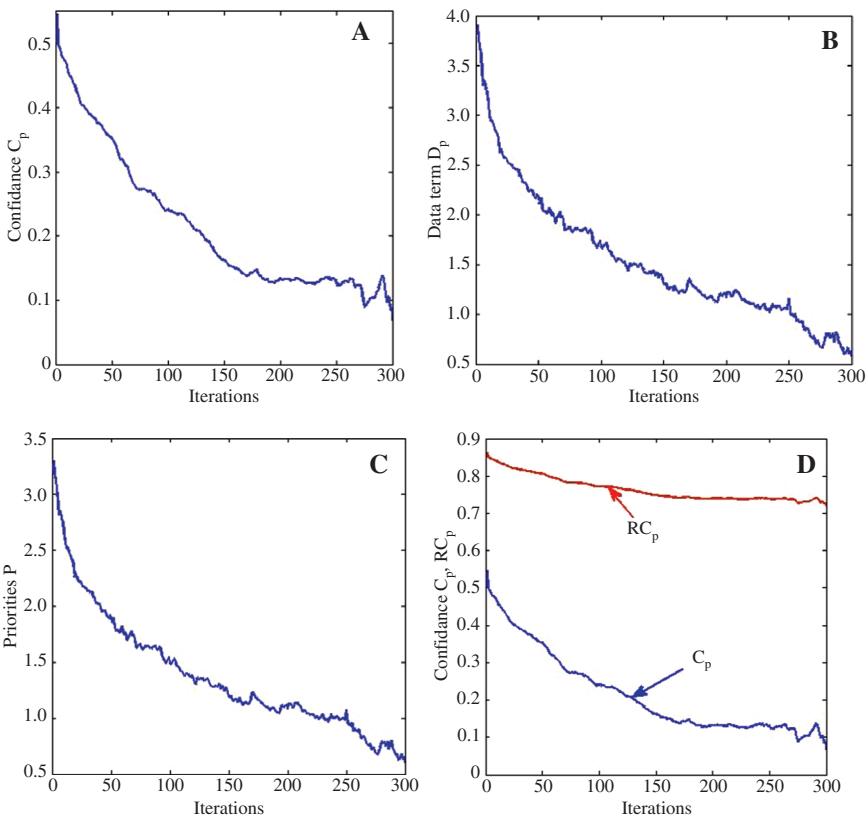

Figure 3. (A) Average Decrease of C_p. (B) Data Term D_p. (C) Priorities. (D) More Linear and Regularized RC_p Over Iterations.

Additional weights can be incorporated to control the contribution of $C(p)$ and $D(p)$ in priorities $P(p)$ for various types of images:

$$P(p)=\alpha\times RC(p)+\beta\times D(p), \quad 0\leq\alpha,\beta\leq 1, \tag{6}$$

with $\alpha+\beta=1$, where $\alpha=0.7$ and $\beta=0.3$.

It is also obvious that the best matching patch will be found mostly near the target patch. Thus, we bound the search area to improve the performance of the algorithm either by $n\times m$ pixels around ψ_p or by the empirical equation

$$\begin{aligned} X_{\min}&=\max\left(0,p_x-\frac{P_w}{2}-W_r-\frac{D_x}{2}\right) \quad \Big| \quad Y_{\min}=\max\left(0,p_y-\frac{P_w}{2}-W_c-\frac{D_x}{2}\right),\\ X_{\max}&=\min\left(I_h,p_x+\frac{P_w}{2}+W_r+\frac{D_x}{2}\right) \quad \Big| \quad Y_{\max}=\min\left(I_w,p_y+\frac{P_w}{2}+W_c+\frac{D_x}{2}\right), \end{aligned} \tag{7}$$

where P_w is the patch width, I_w is the image width, p_x and p_y are the pixel coordinates, D_x and D_y are the extra search margins, and W_c and W_r are the target region dimensions.

We searched the best matching patch $\psi_{q\in\phi}$ of ψ_p in the bounded area by computing the sum squared error (SSE). It is possible that there are multiple best matching patches $\psi_{q\in\phi}$ with the same SSE. To resolve such conflict, we computed the variance *wrt* mean of $\psi_{p\in\phi}$ as

$$M=\frac{\sum(I_{p\in\phi\cap\psi p})}{\#\{p|_{p\in\phi\cap\psi p}\}} \tag{8}$$

$$V=\frac{\sum(I_{p\in\phi-\psi p}-M)^2}{\#\{p|_{p\in\phi-\psi p}\}}, \tag{9}$$

where p defines the size of the set.

2.2 Algorithm of Patch-Based Inpainting

The pictorial representation of the algorithm is shown in Figure 4.

Algorithm:

- Extract the manually selected initial fill front $\partial\Omega$.
- Repeat until done:
  1. Identify the fill front $\partial\Omega^t$. If $\Omega^t=0$, exit.
  2. Compute priorities using Eq. (6).
  3. Find the patch $\psi_{p\in\partial\Omega}$ with the maximum priority.

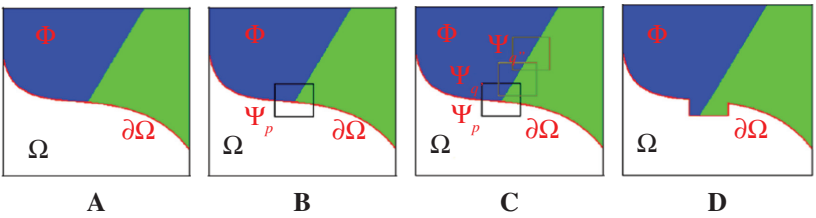

Figure 4. Structure Propagation by Patch-Based Synthesis.
(A) Original Image *I*, with Ω as Region to Fill (Target Region), ∂Ω as the Target Region Boundary (Fill Front), and $\phi = I - \Omega$ as Source Region. (B) ψ_p is the Patch to be Filled, Centered at $p \in \partial\Omega$. (C) The Probable Match of ψ_p is on the Isophote, e.g., $\psi_{q'}$ and $\psi_{q''}$. (d) Unfilled Portion of $\psi_{p\in\Omega}$ is Filled from the Corresponding Best Matching Patch Part $\psi_{q\in\phi-\psi_p}$.

4. Search ϕ for the patch ψ_q that minimizes the error.
5. Copy the image data from the source patch $\psi_{q\in\phi}$ corresponding to $\psi_{p\in\phi}$.
6. Update the confidence term, Eq. (5).

t indicates the iteration number.

3 Implementation Details

3.1 Patch-Based Inpainting With Mask Images

The program starts with a masked degraded image, and the objects to be removed or corrected are masked with a specific color (e.g., [255,0,0]) using any image processing tool (e.g., MS Paint). Isophote computation is done by the image gradients using a gradient rotation of 90° normalized with α [2] as

$$\text{Isophote } (I_p) = \frac{\nabla I_p \text{ rotated by } 90^\circ}{\alpha}, \tag{10}$$

where α is a normalization factor (e.g., 255 for a typical gray-level image and 255×3 for an RGB image). The contour (fill front) of the target region is found by convolving the target region with an LoG edge detector. Normals are calculated as normalized gradients at the fill front because we need to calculate the data term only at the contour. The data term for each pixel on the boundary is computed as

$$D(p) = |I_p| \times N_p. \tag{11}$$

Figure 5. Large-Object Removal.
(A, C) Original Photographs. (B, D) The Region Corresponding to the Foreground Person (Approximately 7 and 22%, Respectively, of the Image) has been Manually Selected and then Automatically Removed. Notice that the Background Structures and Texture have been Synthesized.

The confidence of each pixel is calculated by a patch ψ_p constructed around the pixel on the fill front. Then an array of pixel indices of $\psi_p | p \in \phi$ is found. The confidence of the pixel is determined as

$$C(p) = \frac{\text{Pixels already filled in } \psi_p}{\text{Total pixels in } \psi_p}. \tag{12}$$

Using $C(p)$ and $D(p)$, the patch priority $P(p)$ for the boundary pixel is computed using Eq. (6). Typical values of $\alpha = 0.2, \beta = 0.8$, and $\omega = 0.7$ are used as initial values.

The maximum priority pixel on the fill front is found, and the patch is obtained around it as ψ_p (Figure 4B). A part of $\psi_{q \in \Omega}$ is to be filled at the first iteration of the filling process. A global search for the best matching patch ψ_q is performed in ϕ (Figure 4C). Only the pixels of ψ_p corresponding to the target region will be filled from ψ_q (Figure 4D). The fill region status of new pixels is updated. The confidence values are updated by assigning a confidence value of p to the newly filled pixels.

Figure 6. (A) Original Image. (B) Result of Algorithm [8]. (C) Our Algorithm Result.

If a newly filled region contains the isophote, it should be considered in the next iteration. This method of scanning is new and has better results, as presented in the Results section. The algorithm is iterated until an entire fill region is covered. All experiments are run on a 1.6-GHz AMD Athlon dual-core CPU with 2 GB of RAM.

4 Results

The results of the algorithm of Criminisi et al. [8] were compared with our patch-based inpainting algorithm. Figure 5 indicates the effectiveness of the proposed algorithm for large object removal. Here an algorithm is applied to a variety of color images (Figure 6), which include complex structures and textures. The patch size is set to be greater than the largest texture element or the thickest structure (e.g., edges) in the source region.

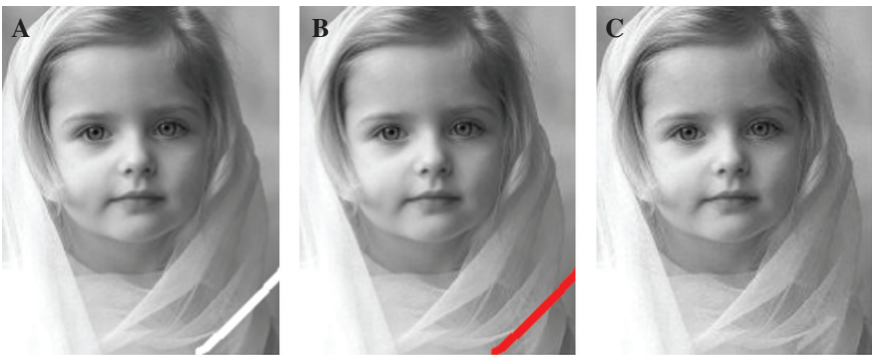

Figure 7. Nice1: Patch Size 9×9.

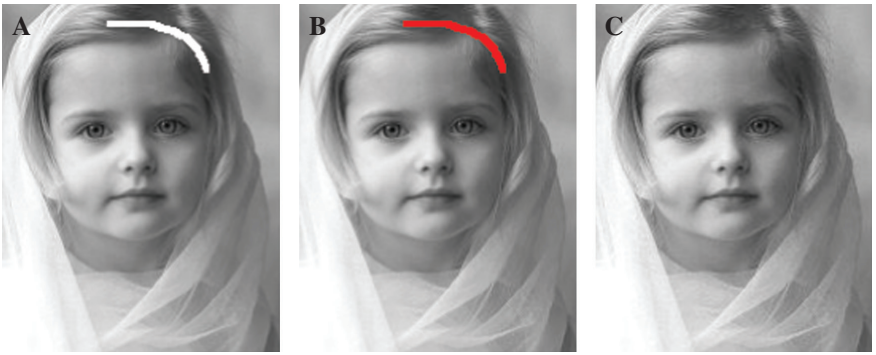

Figure 8. Nice2: Patch Size 9×9.

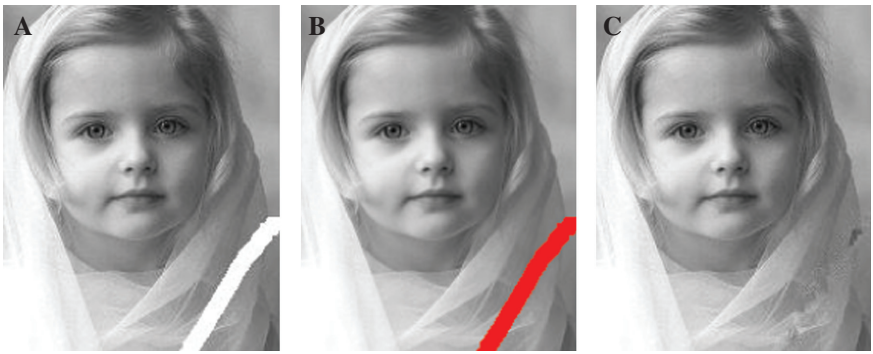

Figure 9. Results with Larger Degradation (Nice3): Patch Size 9×9.

Additional experimental results with synthetic images and photographs are shown in Figures 7–16. In all the results, (a) is the degraded image, (b) is the mask image, and (c) is the result of our inpainting algorithm. For Figures 7–9, the patch

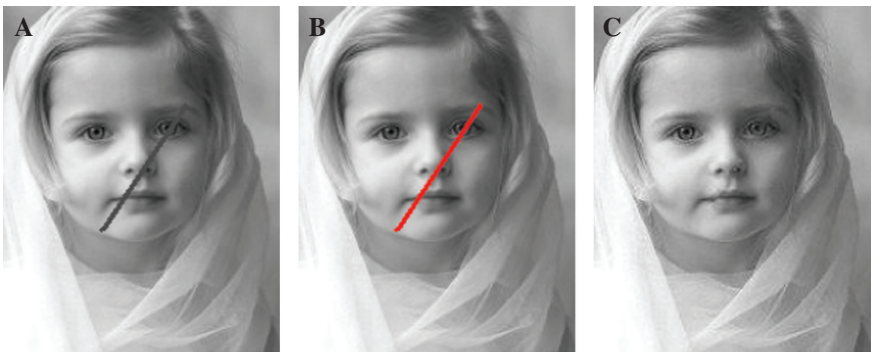

Figure 10. Results with Scratch on Edges (Nice4): Patch Size 7×7.

Figure 11. The Child Image Reconstruction: Patch Size 7×7.

Figure 12. Image Repairing Example (Lenna): Patch Size 9×9.

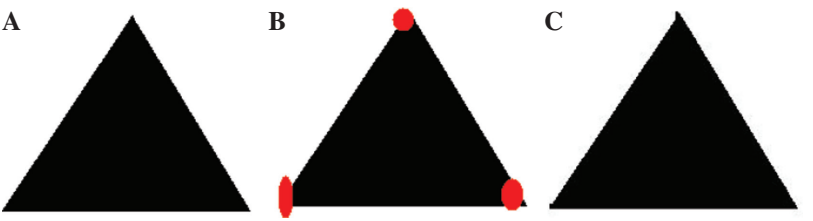

Figure 13. Corner Reconstruction.

Figure 14. Large Object Removal in Photograph: Patch Size 9×9.

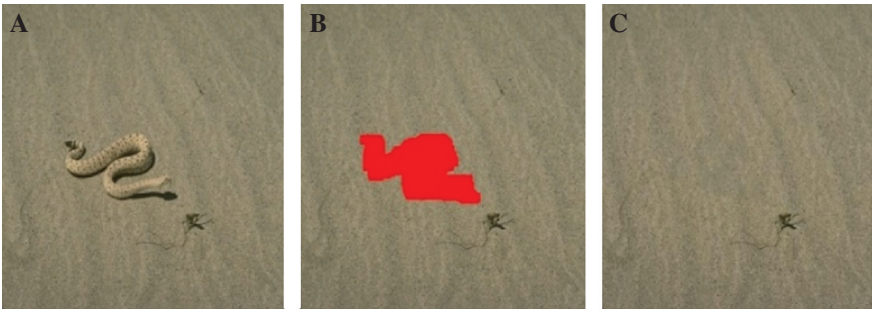

Figure 15. Removing an Object on a Highly Textured Background: Patch Size 9×9.

Figure 16. Curved Structure Reconstruction: Patch Size 9×9.

Table 1. Comparative Results.

Image	Size in pixels		Original and degraded		Original and inpainted		Entropy (×E+10)		
	Image	Fill Area	RMSE	PSNR (dB)	RMSE	PSNR (dB)	Original	Degraded	Inpainted
Nice1	38,720	1124	18.31	22.87	3.887	36.34	−3.49	−3.653	−3.482
Nice2	38,720	376	17.19	23.42	0.96	48.34	−3.49	−3.561	−3.487
Nice3	38,720	88	6.52	31.84	0.73	50.81	−3.49	−3.489	−3.487
Aeroplane	154,401	14,212	58.42	12.78	24.35	17.32	−6.86	−9.63	−7.27
Bungee	63,448	7996	56.67	13.0	34.89	17.27	−1.124	−2.32	−7.41
Child	152,100	1421	3.3	37.75	3.0	38.4	−5.884	−5.889	−5.881

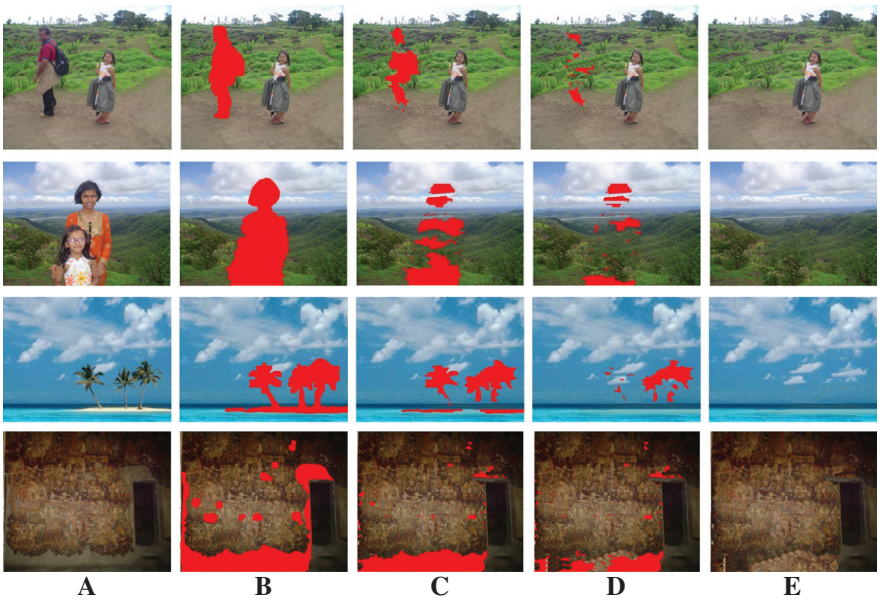

Figure 17. (A) The Original (B–D) are Intermediate Images. (E) The Inpainted Image by our Algorithm.

size is set to 9×9; for Figures 10 and 11, a patch size of 7×7 suffices. Figure 12 shows the distributed fill area. Figure 13 indicates that the corner artifacts result from two isophote joints hidden under the target region. Figures 14 and 15 show large object removal and reconstruction of a relative smooth background. Figure 16 shows the efficacy of our algorithm in curved edge reconstruction. The statistical comparisons of a few representative results in terms of root mean square error (RMSE) and peak signal-to-noise ratio (PSNR) are given in Table 1.

Figure 17 shows the sample frames of a damaged ancient Ajanta painting generated through the iterations of the algorithms for four images. The results in Figure 17E are encouraging.

It should be noted that the time required for a 7×7 patch size is higher compared with that of a 9×9 patch size.

5 Conclusions

This article has presented a modified method for removing large objects from digital photographs. The result is an image in which the selected object has been replaced by a visually reasonable background. This approach uses a patch-based texture synthesis technique modulated by a scheme for determining the fill order of the target region. Pixels maintain a confidence value that, together with image isophotes, influences their fill priority. The technique is capable of propagating both a linear structure and a two-dimensional texture into the target region with a single, simple algorithm. The present studies also indicate that a simple selection of the fill order is necessary and sufficient to handle this task.

This method performs well for large object removal as well as restoration of small scratches. In instances in which larger objects are removed, this method dramatically outperforms earlier works in terms of both perceptual quality and computational efficiency.

Received May 10, 2013; previously published online July 10, 2013.

Bibliography

[1] M. Ashikhmin, Synthesizing natural textures, in: *Proceedings of the ACM Symposium on Interactive 3D Graphics*, pp. 217–226, Research Triangle Park, NC, 2001.

[2] C. Ballester, V. Caselles, J. Verdera, M. Bertalmio and G. Sapiro, A variational model for filling-in gray level and color images, in: *Proceedings of the International Conference on Computer Vision*, pp. 10–16, Vancouver, Canada, 2001.
[3] M. Bertalmio, A. L. Bertozzi and G. Sapiro, Navier–Stokes, fluid dynamics, and image and video inpainting, in: *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 355–362, Hawaii, 2001.
[4] M. Bertalmio, G. Sapiro, V. Caselles and C. Ballester, Image inpainting, in: *SIGGRAP*, pp. 417–424, 2000.
[5] M. Bertalmio, L. Vese, G. Sapiro and S. Osher, Simultaneous structure and texture image inpainting, in: *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Madison, WI, USA, 2003.
[6] R. Bornard, E. Lecan, L. Laborelli and J. -H. Chenot, Missing data correction in still images, in: *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Madison, WI, USA, 2003.
[7] T. F. Chan and J. Shen, Non-texture inpainting by curvature-driven diffusions (CDD), *J. Visual Commun. Image Represent.* **12** (2001), 436–449.
[8] A. Criminisi, P. Perez and K. Toyama, Region filling and object removal by exemplar-based inpainting, *IEEE Trans. Image Process.* **9** (2004), 1200–1212.
[9] S. de Bonet, Multiresolution sampling procedure for analysis and synthesis of texture images, in: *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH*), pp. 361–368, 1997.
[10] I. Drori, D. Cohen-Or and H. Yeshurun. Fragment-based image completion, *ACM Trans. Graph. (SIGGRAPH 2003 Issue)* **22** (2003), 303–312.
[11] A. Efros and T. Leung, Texture synthesis by non-parametric sampling, in: *17th IEEE International Conference on Computer Vision*, pp. 1033–1038, 1999.
[12] A. Efros and W. T. Freeman, Image quilting for texture synthesis and transfer, in: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2001)*, E. Fiume, Ed., pp. 341–346, ACM, New York, 2001.
[13] D. Garber, Computational models for texture analysis and texture synthesis, PhD thesis, University of Southern California, 1981.
[14] P. Harrison, A non-hierarchical procedure for re-synthesis of complex texture, in: *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, Czech Republic, 2001.
[15] J. Heeger and J. R. Bergen, Pyramid-based texture analysis/synthesis, in: *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH*), pp. 229–233, Los Angeles, CA, USA, 1995.
[16] A. Kokaram, R. Morris, W. Fitzgerald and P. Rayner, Interpolation of missing data in image sequences, *IEEE Trans. Image Process.* **4** (1995), 1509–1519.
[17] L. Liang, C. Liu, Y. -Q. Xu, B. Guo and H.-Y. Shum, Real-time texture synthesis by patch-based sampling, *ACM Trans. Graph.* **20** (2001), 127–150.
[18] S. Masnou and J. -M. Morel, Level lines based disocclusion, *Int. Conf. Image Process.* **3** (1998), 259–263.
[19] J. Wu and Q. Ruan, Object removal by cross isophotes exemplar-based inpainting, in: *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 810–813, 2006.