ISSN (Online): 2230-7893
www.IJCEM.org

Image Restoration Using Filling-In Technique

Priyanka Rajesh Gulhane[1], V. T. Gaikwad[2]

[1] Research Scholar, Computer Science and Engineering,
Sipna's College of Engineering & Technology,
Amravati, India
[2]Asst. Prof., Computer Science and Engineering,
Sipna's College of Engineering & Technology,
Amravati, India
gulhane.priyanka@gmail.com, vtgaikwad@rediffmail.com

Abstract

The filling-in of missing information is a very important technique in image processing. While transmission of image if some blocks of image are lost then instead of using common retransmission query protocols, we aim to reconstruct the lost data using correlation between the lost block and its neighbours. Removing a target object and filling the missing regions of an image is the key technology generally applied to image restoration. If the lost block contained structure, it is reconstructed using an image inpainting algorithm, while the same region in the texture image is filled-in with texture synthesis techniques. Depending on the type of missing block, an image inpainting algorithm or texture synthesis technique is applied. Applications of this technique include the restoration of old photographs and removal of superimposed text like dates, subtitles, or publicity. In this paper we proposed a scheme for reconstruction of the lost block from the image. The performance of this method is tested for various images and combinations of lost blocks.

Keywords— *filling-in, texture synthesis retransmission*

1. Introduction

Image processing is a wide area including various applications in it. Since the early days of art and photography, filling-in and inpainting has been done by professional artist. Imitating their performance with semi-automatic digital techniques is currently an active area of research. The filling-in technique the basic idea is to first classify the block as *textured* or *structured* (containing edges), and then fill-in the missing block with information propagated from the surrounding pixels. This information can be automatically detected as in image inpainting and texture synthesis [1], [2], or hinted by the user as in more classical texture filling techniques [3], [4], [5].

In the case of structured blocks, the inpainting algorithm is used, while for textured regions we use texture synthesis techniques. This means that for ordinary images which contain both structure and texture different technique work better for different parts. The switch between the structure and texture is done by analyzing the area surrounding the region to be filled-in, and selecting either a texture synthesis or a structure inpainting technique.

Filling-in missing data in digital images has a number of fundamental applications. They range from removing objects from a scene all the way to retouching damaged paintings and photographs. The basic idea is to fill-in the gap of missing data in a form that is nondetectable by an ordinary observer. In art, this process is called inpainting. Image inpainting provides a means to restore damaged region of an image, such that the image looks complete and natural after the inpainting process. Digital image inpainting mainly aims at filling in missing pixels in an unknown region of an image in a visually plausible way.

Texture is an important aspect of computer graphics, because it allows increasing the realism of images. Textures have been traditionally classified as either regular (consisting of repeated texels) or stochastic (without explicit texels). However, almost all real-world textures lie somewhere in between these two extremes. It can describe a wide variety of surface characteristics such as terrain, plants, minerals, fur and skin. Since reproducing the visual realism of the real world is a major goal for computer graphics, textures are commonly employed when rendering synthetic images. These textures can be obtained from a variety of sources such as hand-drawn pictures or scanned photographs. Hand-drawn pictures can be aesthetically pleasing, but it is hard to make them photo-realistic. Most scanned images, however, are of inadequate size and can lead to visible seams or repetition if they are directly used for texture mapping. Texture synthesis is an alternative way to create textures. Because synthetic textures can be made of any size, visual repetition is avoided. Texture synthesis can also produce tileable images by properly handling the boundary conditions.

2. Related work

Most of the schemes reported in the literature deal with image transmission in error-prone environments. The ensuring reconstruction scheme benefits because, it reduces the need of retransmission. There have been many texture synthesis algorithms proposed over the years. Image Quilting is based directly on texture synthesis [2], so reviewing other algorithms based on that work might provide some insight into possible extensions. Most of these papers, such as [6] use various data structures and search methods to increase the efficiency of the algorithm while maintaining the output quality.

Zhu et. al. model texture as a Markov Random Field and use Gibbs sampling for synthesis. Unfortunately, Gibbs
sampling is notoriously slow and in fact it is not possible to assess when it has converged. Heeger and Bergen [4] try to coerce a random noise image into a texture sample by matching the filter response histograms at different spatial scales. While this technique works well on highly stochastic textures, the histograms are not powerful enough to represent more structured texture patterns such as bricks.

De Bonet [3] also uses a multi-resolution filter-based approach in which a texture patch at a finer scale is conditioned on its "parents" at the coarser scales. The algorithm works by taking the input texture sample and randomizing it in such a way as to preserve these inter-scale dependencies. This method can successfully synthesize a wide range of textures although the randomness parameter seems to exhibit perceptually correct behavior only on largely stochastic textures. Another drawback of this method is the way texture images larger than the input are generated. The input texture sample is simply replicated to fill the desired dimensions before the synthesis process, implicitly assuming that all textures are tilable which is clearly not correct.

E. Chang describe a packetization scheme in which the DCT coefficients array generated by JPEG is grouped such that bursty (consecutive) packet loss during transmission is scattered into a pseudo-random loss in the image domain (i.e., consecutive blocks are rarely lost in the image domain). The ensuing reconstruction scheme benefits because, most frequency components can be recovered from adjacent blocks. However, large bursts may cause the errors to cluster in the image, and reconstruction suffers. It should be noted that the packetization scheme proposed in [7], when used with the reconstruction scheme described in our paper, is expected to further improve and provide satisfactory reconstruction results even for very large bursts.

For this the proposed scheme is in the separation of the lost blocks into different classes, followed by the used of state-of-the-art image filling-in algorithms for textured and structured regions. This is done in a complete automatic fashion and without any side information.

3. Proposed Work

The missing block in image may contain structure or texture. If the lost block contained structure, it is reconstructed using an image inpainting algorithm, while if same region contain texture then it is filled-in with texture synthesis techniques.

The reconstruction of lost blocks can be achieved from the following steps-

a) Classification of the I/P lost block of images into texture and structure;

b) Texture synthesis technique will be used for textured block;

c) Image inpainting algorithm will be used for structured block;

The flow chart of the proposed work is as follows-

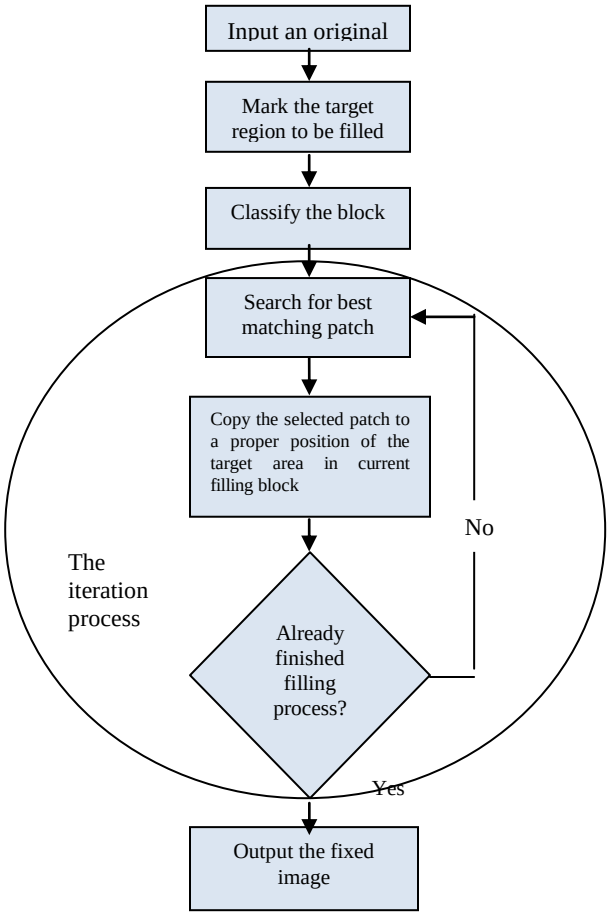


Fig.1 Flow chart of proposed work

3.1 Classification of the Image Block

In the reconstruction process we first classify the lost blocks into texture or structure. The decision about the type of lost block is taken at the receiver by anlysing the region surrounding the lost block. Lost blocks are not included in the querying process. To determine whether or not a block has texture (or noise), we use a simple coarseness measure given by the number of local extrema in the neighborhood of the lost block. The number of local extrema are nothing but the pixels which are local row extrema as well as local column extrema.

Using the method of [9], the number of local extrema in a window of side 8×8 is given by

$$n = \frac{s^2 (UB+LB)}{2}$$

Where UB and LB are respectively the upper and lower bounds for texture coarseness and are selected by the user. These coarseness values vary from 0 i.e. no extrema to 1 i.e. all pixels in the selected window are extrema. We have used UB = 0.16 and LB = 0.04 as suggested in [9]. In this implementation, s=8, giving n=6.4. Thus, if a 8×8 block has fewer than n extrema, then it is classified to have structure, else is considered to contain texture.

This technique is applied in the immediate neighbourhood of the lost block for each available block of 8×8 pixels. Even if a single block from its neighbourhood contains structure, we first consider a decision in favour of structure. However, since reconstruction is our primary goal, this criterion alone might be insufficient, as we illustrate now. Consider for example that we have lost a block containing an edge between two textured regions. The edge between two regions is certainly an expression of structure, and needs to be given precedence over texture even if the block in question has more coarseness than the necessary coarseness. In the case of a block containing an edge as texture, we would not be able to reconstruct the edge later, as will become clear after examining the texture synthesis algorithm.

To overcome this limitation, we impose an additional constraint as follows. We consider the 8-neighborhood of a 8×8 block and calculate differences between the average values of the blocks on opposite sides of the centre block. If the four resulting differences are above a threshold, we decide that an edge does indeed pass through the textured block. We then designate the block as structure, notwithstanding its high coarseness. This simple additional constraint has provided a correct classification in all tested images.

3.2 Texture Synthesis Technique

From the earlier classification, we conclude that when a block is classified as having texture, the entire 8-neighborhood of that block has texture. The missing block is then filled-in (reconstructed) with the texture from its surrounding.

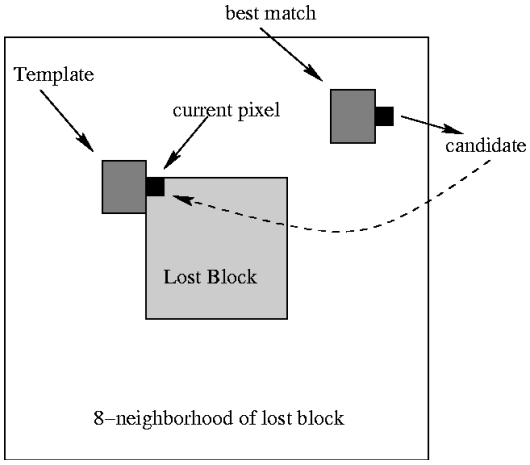


Fig. 2 Texture synthesis procedure.

Let the region to be filled be denoted by Ω. The lost block will now be filled, pixel by pixel, in a raster fashion. Let I_t be a representative template touching the left of a pixel p(i, j) ϵ Ω. We proceed to find a I_t from the available neighborhood, such that a given distance $d(I_t, \hat{I}_t)$ is minimized. As per [5], d is a normalized sum of squared differences (SSD) metric. Once such a, $\hat{I}_t$ is found, we choose the pixel to the immediate right of $\hat{I}_t$, as our candidate for p(i, j) ϵ Ω. For stochastic textures, the algorithm selects at random one of the pixels neighboring $\hat{I}_t$.

The template $\hat{I}_t$ can be a simple seed-block of 3 × 3 pixels as shown in Fig. 2. Then, of all possible 3 × 3 blocks in the 8-neighborhood, the one with the minimum normalized SSD is found and a pixel to its right is copied into the current pixel in the lost block, as shown. This algorithm is considerably fast when using the improvements in [6], [10].

3.3 Image Inpainting

Structure in an image can be an edge between two regions or a deterministic change in color or gray value. When the block classification algorithm detected a structured block, this is reconstructed using the digital inpainting procedure.

Once again let Ω be the region to be filled in (inpainted) and δΩ be its boundary. The basic idea in

ISSN (Online): 2230-7893
www.IJCEM.org

inpainting is to smoothly propagate the information surrounding Ω in the direction of the isophotes entering δΩ. Both gray values and isophote directions are propagated inside restored the region. Denoting by *I* the image, this propagation is achieved by numerically solving the partial differential equation (t is an artificial time marching parameter)

$$\frac{\partial I}{\partial t} = \nabla(\Delta I) \cdot \nabla^{\perp} I$$

where ∇ , Δ , and $\nabla\perp$ stand for the gradient, Laplacian, and orthogonal-gradient (isophote direction) respectively. This equation is solved only inside Ω, with proper boundary conditions in δΩ for the gray values and isophote directions.

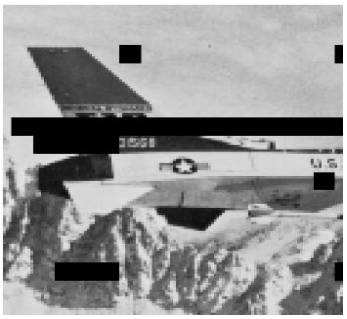
Note that at steady state, $(\partial I/\partial t) = 0$, and $\nabla(\nabla I).\nabla\perp I = 0$. This means that ∇I is constant in the direction $\nabla\perp I$ of the isophotes, thereby achieving a smooth continuation of the Laplacian inside the region to be inpainted.

Transmitted Image

Received Image

Fig. 3 Image with low loss.

Transmitted Image



Received Image

Fig.4 Image with drastic loss.

Since we have no control over the fading channel, there is no prior information about the relative locations and number of blocks that can be lost in the process. We present various examples ranging from low to drastic losses of image information, and demonstrate our proposed technique to restore the lost information. Figs. 3–4 show the results of loss.

We can make the following assumptions.

a) When single blocks are missing from the image, they are satisfactorily reconstructed from the surrounding context. Here the reconstructed image will be almost identical to the original one, as expected from the algorithms used for filling-in.

b) When a few contiguous blocks are missing, the algorithm still reconstructs the blocks so as to be visually unrecognizable from the original.

c) As a drastic condition, if an entire line is missing, then, a good reconstruction is not always possible.

In general, when feature sizes are smaller than 8×8 pixels or are totally covered by the missing line, it will be impossible to reconstruct the image correctly. As shown in Fig. 4. In such cases, we will be forced to request retransmission of the lost block (or an error block). Such instances will become increasingly rare when the image

resolution is increased. Further, if the packetization scheme is used, then it is extremely rare to lose an entire line in the image domain. In that case, only independent lost blocks would need to be reconstructed from their neighborhood, and as indicated by Fig.3, the above algorithm restores isolated blocks reliably.

4. Conclusions and Future Directions

In this paper we have proposed a scheme for reconstruction of lost blocks in the image. We assume that as long as the features in the image are not completely lost, they can be satisfactorily reconstructed using a combination of computationally efficient image inpainting and texture synthesis algorithms. This eliminates the need for retransmission of lost blocks. We have tried to use image-dependent information, i.e., texture and structure, to enhance the performance of image.

In a more general setting, the extension of the approach presented here, to color data needs to be investigated. Since the missing blocks in the different channels need not be in the same image position, information from different channels can be used in the block classification and reconstruction. Adding this to the current neighbouring information used is expected to improve even further the quality of the results.

5. References

M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image inpainting,” in *Comput. Graph. (SIGGRAPH 2000)*, July 2000, pp. 417–424.

Alexei A. Efros and Thomas K. Leung,” Texture Synthesis by Non-parametric Sampling,” in *IEEE Int. Conf. Computer Vision*, Corfu, Greece, Sept. 1999.

J. S. De Bonet, “Multiresolution sampling procedure for analysis and synthesis of texture images,” in *Proc. ACM SIGGRAPH*, July 1997.

David J. Heeger and James R. Bergen,” Pyramid-Based Texture Analysis/Synthesis,” in *Computer Graphics (SIGGRAPH 1995)*, July 1995.

Simoncelli and J. Portilla, “Texture characterization via joint statistics of wavelet coefficient magnitudes,” in *Proc. 5th IEEE Int. Conf. on Image Processing*, 1998.

L. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in SIGGRAPH 2000.

Chang, “An image coding and reconstruction scheme for mobile computing,” in *Proc. 5th IDMS*, Oslo, Norway, Sept. 1998, pp. 137–148.

Z. Alkachouch and M. Bellanger, “Fast dct-based spatial domain interpolation of blocks in images,” *IEEE Trans. Image Processing*, vol. 9, pp.729–732, Apr. 2000.

K. Karu, A. K. Jain, and R. M. Bolle, “Is there any texture in the image?,”*Pattern Recognit.*, vol. 29, no. 9, pp. 1437–1446, 1996.

Marquina and S. Osher, “Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal,” *SIAM J. Sci. Comput.*, vol. 22, pp. 387–405, 2000.