Abstract

In this paper we propose a new approach to handle the problem of restoration of grayscale textured images. The purpose is to recover missing data of a damaged area. The key point is to decompose an image in its bit-planes, and to process bits rather than pixels. We propose two texture synthesis methods for restoration. The first one is a random generation process, based on the conditional probability of bits in the bit-planes. It is designed for images with stochastic textures. The second one is a best-matching method, running on each bit-plane, that is well suited to synthesize periodic patterns. Results are compared with a state-of-the-art restoration algorithm.

1. Introduction and previous works

Filling-in gaps in a digital image, often known as digital inpainting, is one of the most active field in image processing research. Restoration of damaged or unknown areas in an image is an important topic for applications as: image coding (e.g. recovering lost blocks); removal of unwanted objects (e.g. scratches, spots, superimposed text, logos); video special effects; 3D texture mapping. There are different main approaches for a filling-in problem in literature: PDE (Partial Differential Equation) methods, and constrained texture synthesis.

PDE methods[2][3] give impressive results with natural images but introduce blurring, that is more evident for large regions to inpaint. They are computationally expensive and not suitable for textured images.

Texture synthesis methods reconstruct an image from a sample texture. For inpainting purposes, region to fill-in is the area into which synthesize the texture, and information to replicate is taken from the surrounding pixels. Most of these methods use Markov Random Fields as theoretical model[5] to represent a texture. That is, for each pixel, color (or brightness) probability distribution is determined by a limited set of its surrounding pixels.


We propose a new approach to recover damaged information in textured images. Images are processed within a simple domain, the bit-plane representation. Two texture synthesis methods are proposed for the restoration problem. The first one is an improvement of our previous proposed method[1], which had been tested on (poorly textured) images from a photographic archive of digitized old prints. The second one is a new algorithm, designed for textures with a periodic pattern. The purpose is to fill-in gaps, with a generic shape, of an image using surrounding information.

2. The bit-plane representation

Bit-plane slicing is a well known technique used to represent the content of a grayscale image. It is mostly used for application in the field of digital watermarking [12][17] and image compression [8][11]. Garcia et al.[9] proposed a method based on bit-plane splitting to classify greyscale textured images.

The key point our two proposed methods is to observe features of a damaged image in a simple domain, the bit-plane representation. Image is split in
Figure 1 Image bit-plane decomposition. (a) original image, (b-c) most significant, (d-e) less significant bit-planes. Most significant bit-planes are more structured than less significant ones. Lower planes are quite similar to pure noise.

its bit-planes, with a bit-plane slicing decomposition, and Gray-coding is applied, to decorrelate information between different planes. Both the proposed restoration methods work with bits in the bit-plane space, rather than with pixels of the image. Planes are processed from the most significant to the less, and at each step restoration depends on the previously restored planes. They cannot be processed separately, since annoying artifacts would be visible into the reassembled image. Gray coding helps to decorrelate planes, but is not enough. Many are the advantages in terms of efficiency:

- bit information can be stored in a first step (analysis) and used in the second step (synthesis) (see subsection 4.1). To our knowledge, none of the related works proposed a method to store information about pixel, because it is an hard task, both for memory usage and access time problems. Typically a search for the needed information is recomputed at each step of the restoration process, with a waste of execution time.
- working with bits is faster and simpler than working with pixels. At each step the output of the restoration process is simply a binary value (or mask) instead of a pixel value (or mask);
- since most part of information is stored in the most significant planes, lower planes can be processed roughly, (e.g. using smaller windows), speeding-up the process, without losing quality in the restored image. Furthermore, it has been shown[1] that, for natural damaged images, with no superimposed damage, defects are not visible in the lower planes. Less significant planes therefore can be not processed, speeding-up the execution time of the algorithm.

3. Restoration methods

Two texture synthesis based approaches are discussed in next sections. Our texture model is based on the Markov Random Field theory[5], since it has proven to be satisfactory in representing a wide set of texture types. We consider textures as instances of a stationary random model, in which each pixel is statistically determined by its neighborhood.

Both the two approaches don’t focus on automatic damage detection. The user must select the area to restore, to create a binary matrix, in which all the pixels are labeled as good or damaged, used as an input for the algorithms.

The next two sections will provide a detailed description of the two proposed methods.

4. The conditional random generation method

The first proposed method is a generative process, based on bits statistics in the bit-planes. Once the image is decomposed in the bit-plane representation, two are the steps of the algorithm:

- Information analysis
- Reconstruction

An evaluation of the computational cost of this algorithm is given in the subsection 4.3.

4.1. Information Analysis

The purpose of this step is to build a dictionary to store uncorrupted information, which will be used in the reconstruction step.
A square window $W_N$ (where $N$ is the window size set by the user) runs along each bit plane. Bit-planes are processed from the most significant to the less. For each undamaged bit in a bit-plane, an index is created with the scan-ordered bit sequence inside the window $W_N$. Similarly, another index is created with bits in the previous significant bit plane, with an $M$-size square window set in the same position, and added as a header to the first index:

$$k(x,y) = \left[ \sum_{(i,j)\in w^{m}} b^i(x,y) \cdot 2^j \right] \cdot 2^{k_N} + \left[ \sum_{(i,j)\in w^{2}} b^i(x,y) \cdot 2^j \right]$$

(1)

where $b^i(x,y)$ are bits from the current bit-plane, $b^{i+1}(x,y)$ are bits from the previous significant one. The frequencies of these sequences into the bit-planes are stored in a histogram, which is our “dictionary”. Each value is an estimation of the a posteriori probability of a bit sequence in a $i$-plane, conditioned by the corresponding sequence in the previous $(i+1)$-plane. According to the Markov Random Field hypothesis, we suppose that this estimation is equal to the conditional probability value:

$$H(i,k) = P(W_{N_i}^i | W_{M_i}^{i+1}).$$

(2)

The most significant plane is processed as a special case, with no contribution from a previous plane.

### 4.2. Reconstruction

According to the 2D-Wold decomposition model for homogeneous random fields[14][16] textures can be decomposed into a deterministic and a purely indeterministic components. The most important features for human texture perception are: periodicity, directionality and randomness. Two competing processes work to reproduce these features from the global image into the damaged area: a bit-by-bit constrained random generation process, which aims to reproduce texture directionality and randomness of the global image, and a patching process, to replicate texture periodicity.

As a preliminary remark, note that results are strongly affected by the order into which pixels (or bits) are synthesized, because it sets the neighborhood used to reconstruct the damaged area. With a simple scan order the restoration process tends to reproduce up-to-down left-to-right diagonal shapes. Our algorithm processes bits along a direction that depends on image average gradient vector. This solution helps us to reconstruct the natural bias of the image.

The reconstruction phase is the dual process of the dictionary building process. A $N$ square window runs on the damaged area of each plane. As in the previous step, planes are processed from the most significant to the less. For each damaged bit of a plane, the corresponding window will contain uncorrupted, corrected and damaged bits. A $M$ square window is considered in the previous plane, at the same position. The whole information is known for this window (bits are either undamaged or corrected).

The bit-by-bit generation process computes the probability that the central bit of the window is 1 or 0, given the known neighbor bits in the plane and the bits in the previous plane. The statistics of each of the submasks of a window can be computed building up those of all the possible statistics of the windows which share that submask:

$$p(W_{N_i}^i | W_{M_i}^{i+1}, b_z = 0) = \sum p(W_{N_i}^i | W_{M_i}^{i+1}, b_z = 0)$$

(3)

$$p(W_{N_i}^i | W_{M_i}^{i+1}, b_z = 1) = \sum p(W_{N_i}^i | W_{M_i}^{i+1}, b_z = 1)$$

(4)

The two statistics we are looking for:

$$S'_i(x,y) = \sum_{p=0}^{2^{m-1}} H[i,B_p(x,y)] = p(W_{N_i}^i | W_{M_i}^{i+1}, b_z = 0)$$

(5)

$$S''_i(x,y) = \sum_{p=0}^{2^{m-1}} H[i,W_p(x,y)] = p(W_{N_i}^i | W_{M_i}^{i+1}, b_z = 1)$$

(6)

where $H[i,k]$ is the dictionary built in the analysis phase, $N_D$ is the number of the damaged bits in the window, $B_p$ is the index for the sequence with a “black” (zero) central bit in the window, and $W_p$ is the sequence with a “white” (one) central bit. $b_z$ is the central bit of the mask in the $i$-plane. Both of these indexes contain bits from the $W_N$ submask.

The next step is a random generation, conditioned by the statistics computed in eq.5 and eq.6, in order to choice which information (0/1) to put in the central position of the window. The two statistics are weighted by weights that depend on an user-defined parameter $\alpha$:

$$P_0 = w_0 \frac{S_0}{S_0 + S_1}, P_1 = w_1 \frac{S_1}{S_0 + S_1}.$$

(7)

$$w_0, w_1 = \max(P_0, P_1), \alpha$$

By setting $\alpha$ close to 1, this process is the same as a random process with the two probabilities:

$$P_0 = \frac{S'_i(x,y)}{S'_i(x,y) + S''_i(x,y)}, P_1 = \frac{S''_i(x,y)}{S'_i(x,y) + S''_i(x,y)}.$$

(8)

which fits for synthesizing highly stochastic textures. As $\alpha$ increase, the bit value is chosen as the central
bit of the most frequent window with that surrounding conditions. That is suitable for strongly oriented textures. In this way our method can control the randomness and directionality of the generated texture.

To avoid the “growing garbage” problem, if no statistics match the current sequence in the dictionary, a random generation process is used with the following probabilities:

\[ P_0 = P \{ b'(x, y) = 0 \ | \ b'^{-1}(x, y) \} \]
\[ P_1 = P \{ b'(x, y) = 1 \ | \ b'^{-1}(x, y) \} \]

At the same time, a second competing process works to propagate global texture features into the area to restore. A patching process aims to reproduce texture periodicity. For each damaged bit, the two most frequent sequences (one with 0 as its central bit, one with 1), which share the known-bits submask, are extracted from the dictionary:

\[ W_{max}^i(x, y) = \arg \max P \{ W_j^i \ | \ W_{max}^i, b = 0 \} \]
\[ W_{max}^i(x, y) = \arg \max P \{ W_j^i \ | \ W_{max}^i, b = 1 \} \]

If one of the statistics is much greater than the other, the bit-by-bit generation process is disabled, for the current step, and the whole window is filled in with the most frequent sequence. The activation threshold of this process, that is what we mean for “much greater”, is set by an user defined parameter. As we discussed in this section, less significant bit-planes have a more random global structure. Therefore patching is useless or harmful to process these planes, and it is disabled. Filling-in the whole window rather than bit-by-bit extremely speeds up the execution time, and it helps in replicating texture periodicity, if it is at a scale either equal or smaller than the window size.

After all planes are restored, bit planes are merged to reconstruct the whole image, and a soft edge-preserving smooth filter is applied to remove the residual high-frequency noise due to this reassembling phase.

4.3. Computational Cost

Computational cost depends on damaged area size and on the windows size:

\[ O(n - d) + O(d \cdot 2^{S-T}) \]
\[ S = N \times N + M \times M \]

where d is damaged pixels number, n is image size, and T is the table index size. The first term of eq. 12 results from the dictionary building phase. It also depends on windows size. The second term is the computational cost of the reconstruction phase. Exponential term is due to the structure we use to store information in the analysis step. Our dictionary is stored in a hash table, with collision lists, which is the best solution to speed-up the access time. T is the table size. If d<<n and windows are small, first term is predominant and computational cost is O(n).

Increasing M, N and d, computational cost becomes exponential in the worst case, that is much far from the real execution time measured with our experiments.

5. The best matching method

The second proposed algorithm is a best matching method. Many are the contact points between the two methods:
- they both work on bit-planes;
- they need an input binary matrix, with the position of the damaged bits;
- planes are processed from most significant to the less;
- for each plane, bits are processed along a direction that depends on the average gradient vector.

The algorithm is much simpler than the first one. For each bit-plane, a window \( W_N \) runs along the damaged area, and a corresponding window \( W_M \), set in the same position, runs along the previous significant plane. For each damaged bit in each bit-plane the algorithm creates a word \( w(x, y) \) of bits, as in eq.1, using both information from the current and the previous significant bit-plane. The most similar word, according to the Hamming distance, is searched in a neighborhood (neighborhood size is set by the user). Only known bits are considered for matching. Finally, bits from the best matching word are used to replace the unknown bits of the word \( w(x, y) \). Once all the planes are restored, they are merged to reconstruct the whole image. Computational cost is linear with neighborhood and window sizes, and with number of damaged bits.

6. Experimental results

Tests had been made on over 40 640x640 grayscale images from the Brodatz texture set. Both stochastic and periodic texture are used for tests. Each image is arbitrarily damaged to create an area to fill-in. The algorithms have been implemented in ANSI-C, and executed on an Intel Core Duo PC (1.83 GHz, 2 GB RAM).

Figure 2 shows some results obtained with our first algorithm, compared with those obtained with the Criminisi inpainting algorithm[4]. Both visual and numerical comparison are provided. Visual comparison shows that our results are very similar to those obtained with Criminisi algorithm. For a quantitative evaluation of the results, we measured a set of significant statistical parameters. No
remarkable differences in statistical features measured for the two methods (with respect to the parameters of the original image). We measured only some difference in the S/N parameter for small area to fill. This can be explained by considering that the Criminisi algorithm is based on a patching method. Our method, on the other hand, is one or two order of magnitude faster than the Criminisi method, depending on the damaged area size. Execution time is about 1 sec, for stochastic texture and small holes, and rises up to 5-6 minutes, for highly-structured textures and large-sized holes, processed with larger-sized masks.

Figure 3 shows some results for the second proposed method, compared to those obtained with the Criminisi algorithm. Visual comparison show that our method gives impressive results, comparable to those obtained with Criminisi. For the quantitative evaluation, very little differences are measured in statistical parameters, but a higher signal-to-noise ratio gain. Execution time is about a half of the time measured to execute the Criminisi algorithm.

7. Remarks and limitations

Experimental results showed that the two methods are complementary, in the sense of the typology of textures they are suited for.
The conditional random generation process works well with stochastic textures. The most evident limitation of this approach is the window size. There are two problems with large-sized windows: the larger the window, the higher the execution time is; the larger the window, the less consistent the statistics stored in the dictionary is. Note that to create consistent statistics, hole size must be much lower than image size, which is usually the case in real-world images. Therefore tests have been made with a maximum window size of 7x7. This is not a problem for processing stochastic texture (a 3x3 window performs well). Textures that have periodicity in larger scale are harder to reconstruct.

Note that only the most significant bit-planes need larger windows. Lower planes have more random structure, and if higher planes are well-reconstructed, they can be restored using smaller windows.

The best matching method gives impressive results with textures with periodic patterns, no matter what the scale of the periodicity is. It takes the most similar information outside the damaged area, and replicates it into the gap. That’s the reason why it works well with periodic patterns. For stochastic textures, note that using bits rather than pixels, the probability to have more than one best-matching word is higher, above all for small-sized windows. Therefore, the way to decide which best word to

<table>
<thead>
<tr>
<th>stats</th>
<th>original</th>
<th>damaged</th>
<th>our method</th>
<th>Criminisi</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>38,0926</td>
<td>37,5912</td>
<td>37,9622</td>
<td>38,1038</td>
</tr>
<tr>
<td>σ</td>
<td>32,5217</td>
<td>32,5940</td>
<td>32,5307</td>
<td>32,5152</td>
</tr>
<tr>
<td>s</td>
<td>3,6218</td>
<td>3,5787</td>
<td>3,6185</td>
<td>3,6231</td>
</tr>
<tr>
<td>k</td>
<td>18,5613</td>
<td>18,4199</td>
<td>18,5372</td>
<td>18,5789</td>
</tr>
<tr>
<td>S/N</td>
<td>15,0393</td>
<td>20,0748</td>
<td>18,0450</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td>5315</td>
<td>(1,3%)</td>
</tr>
<tr>
<td>t(s)</td>
<td></td>
<td></td>
<td>24,39</td>
<td>39,62</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stats</th>
<th>original</th>
<th>damaged</th>
<th>our method</th>
<th>Criminisi</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>165,2731</td>
<td>165,7008</td>
<td>165,1780</td>
<td>165,2943</td>
</tr>
<tr>
<td>σ</td>
<td>42,9374</td>
<td>43,2738</td>
<td>42,9035</td>
<td>42,9339</td>
</tr>
<tr>
<td>s</td>
<td>-1,3630</td>
<td>-1,3163</td>
<td>-1,3665</td>
<td>-1,3634</td>
</tr>
<tr>
<td>k</td>
<td>5,1276</td>
<td>5,0862</td>
<td>5,1336</td>
<td>5,1276</td>
</tr>
<tr>
<td>S/N</td>
<td>15,9187</td>
<td>28,3516</td>
<td>28,9309</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td>2293</td>
<td>(0,99%)</td>
</tr>
<tr>
<td>t(s)</td>
<td></td>
<td></td>
<td>16,51</td>
<td>32,61</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stats</th>
<th>original</th>
<th>damaged</th>
<th>our method</th>
<th>Criminisi</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>147,7488</td>
<td>146,8935</td>
<td>147,6906</td>
<td>147,7991</td>
</tr>
<tr>
<td>σ</td>
<td>64,0724</td>
<td>64,8224</td>
<td>64,0326</td>
<td>64,0907</td>
</tr>
<tr>
<td>s</td>
<td>-0,3168</td>
<td>-0,3307</td>
<td>-0,3179</td>
<td>-0,3171</td>
</tr>
<tr>
<td>k</td>
<td>1,6361</td>
<td>1,6833</td>
<td>1,6392</td>
<td>1,6361</td>
</tr>
<tr>
<td>S/N</td>
<td>14,2774</td>
<td>22,0735</td>
<td>20,2075</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td>3190</td>
<td>(0,78%)</td>
</tr>
<tr>
<td>t(s)</td>
<td></td>
<td></td>
<td>23,42</td>
<td>42,02</td>
</tr>
</tbody>
</table>

Figure 3. (a-c) corrupted images (D34, D1, D35 from the Brodatz set, with superimposed damages); (d-i) restored images (detail from the reconstructed zone) with our method (d,f,h) and with Criminisi inpainting algorithm (e,g,i). Statistical parameters and signal to noise ratio are provided to evaluate the quality of the results of the two methods. Execution time is measured to compare efficiency.
choice is a critical point. The risk is to choose always the same sequence, and to have excessive repetition of the patches, that is an evident artifact in stochastic textures. A method to select the best word in case of more than one candidate is an open issue.

8. Conclusions and future works

Working with bits is faster and simpler than processing pixels. This is the key point of the presented approach. Image is decomposed in its bit-plane representation. Two methods, working on bit-planes, are proposed to process a wide set of texture types: a conditional random generation process and a best matching method. The first method gives better results with stochastic textures. The second one works well with periodic patterns. Experimental results showed that efficiency is improved, in respect of related works, with no loss in visual quality.

In the future we plan to study a method to automatically select the best method for an image, and to eliminate dependence from the user-defined parameters. Texture features could be estimated during a pre-analysis phase, and parameters suggested for the restoration process.

Acknowledgements

This work has been partially funded by the MIUR (Italian Ministry of Education, University and Research) project FIRB 2003 D.D. 2186–Ric, December 12th 2003.

References


