International Journal of
Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.03,Issue.11
June-2014,
Pages:2355-2363

www.semargroup.org,
www.ijsetr.com

Image Filling By Using Texture Synthesis

SREEKANTH MUNDURU[1], MADHUSUDHANA RAO KOTHARI[2], PRIYANKA R[3]
[1]Raghu Engineering College, Modavalasa, Visakhapatnam, Andhrapradesh, India.
[2]Tata Power SED, Electronics City, Hosur, Road, Bangalore, Karnataka, India.
[3]RV College of Engineering, Bangalore, Karnataka, India.

Abstract: A new algorithm is proposed for removing large objects from digital images. The challenge is to fill in the hole that is left behind in a visually plausible way. In the past, this problem has been addressed by two classes of algorithms: (i) "texture synthesis" algorithms for generating large image regions from sample textures (ii) "inpainting" techniques for filling in small image gaps. The former has been demonstrated for "textures" repeating two-dimensional patterns with some stochasticity; the latter focus on linear "structures" which can be thought of as one-dimensional patterns, such as lines and object contours. This project presents a novel and efficient algorithm that combines the advantages of these two approaches. We first note that exemplar-based texture synthesis contains the essential process required to replicate both texture and structure; the success of structure propagation, however, is highly dependent on the order in which the filling proceeds. We propose a best-first algorithm in which the confidence in the synthesized pixel values is propagated in a manner similar to the propagation of information in inpainting. The actual colour values are computed using exemplar-based synthesis. Here the simultaneous propagation of texture and structure information is achieved by a single, efficient algorithm. Computational efficiency is achieved by a block-based sampling process. A number of examples on real and synthetic images demonstrate the effectiveness of our algorithm in removing large occluding objects as well as thin scratches. Robustness with respect to the shape of the manually selected target region is also demonstrated. Our results compare favorably to those obtained by existing techniques.

Keywords: Filling, Texture, Synthesis, Inpainting.

I. INTRODUCTION

Region filling is filling the Lacuna region (the region left after removal of the object) left after object removal with reasonable feeling for human visual system (reference) .The input for our proposed region-filling algorithm is an image with a Lacuna region and our proposed algorithm can infer the Lacuna region from the remaining part of the image (the source region). (eg: Ashikmin, 2001) The output image after region filling is shown in Fig. 9(c). The "valid" pixels of the source region serve as examples for filling the Lacuna region. In computer vision, texture synthesis algorithms generate a large area of similar texture from sample texture or fill the lost region with input texture[7]. Image inpainting algorithms are used to repair the scratches or cracks of photographs and paintings[1]. Generally speaking, texture synthesis is applied to problems of single texture and image inpainting is used in general images with multiple textures[9].

A. Review of Texture Synthesis

The techniques are classified mainly into three categories. The first category is the synthesis of texture by simulating the physical generation process. The second category is the derivation of a parametric model by analyzing the input texture and synthesizing the output texture. However, these approaches are not able to capture the local features of texture. The third category of texture synthesis algorithms is the generation of the output texture by reproducing the sample texture. This is usually accomplished by synthesizing the pixels with the highest level of similarity to the Lacuna region. Texture synthesis has a variety of applications in computer vision, graphics, and image processing. An important motivation for texture synthesis comes from texture mapping. Texture images usually come from scanned photographs, and the available photographs may be too small to cover the entire object surface.

In this situation, a simple tiling will introduce unacceptable artifacts in the forms of visible repetition and seams. Texture synthesis solves this problem by generating textures of the desired sizes. Other applications of texture synthesis include various image processing tasks such as occlusion fill-in and image/video compression. The texture synthesis problem may be stated as follows. Given an input sample texture I_{in}, synthesize a texture I_{out} that is sufficiently different from the given sample texture, yet appears perceptually to be generated by the same underlying stochastic process. In this work, we use the Markov Random Field (MRF) as our texture model and assume that the underlying stochastic process is both local and stationary. We choose MRF because it is known to accurately model a wide range of textures.

In recent years, a number of successful texture synthesis algorithms have been proposed in graphics and vision. Motivated by psychology studies, Heeger and Bergen [1995] developed a pyramid-based texture synthesis algorithm that approximately matches marginal histograms of filter responses. High-quality texture can be synthesized in real-time on a midlevel PC. A key ingredient of our patch-based sampling algorithm is a sampling scheme that uses texture patches of the sample texture as building blocks for texture synthesis. The patch-based sampling algorithm is fast. Figure shows an example produced by our algorithm. After spending 0.6 seconds analyzing the input sample, our algorithm took 0.02 seconds to synthesize this texture on a 667 MHz PC. The patch-based sampling algorithm works well for a wide variety of textures ranging from regular to stochastic.

The patch-based sampling algorithm is an extension of our earlier work on texture synthesis by random patch pasting. Praun has successfully adapted this patch-pasting algorithm for texture mapping on 3-D surfaces. Unfortunately, these patch-pasting algorithms suffer from mismatching features across patch boundaries. The patch-based sampling algorithm, on the other hand, avoids mismatching features across patch boundaries by sampling texture patches according to the local conditional MRF density. Patch-based sampling includes patch pasting as a special case, in which the local PDF implies a null statistical constraint. Patch-based sampling is amenable to acceleration and the fast speed of our algorithm is partially attributable to our carefully designed acceleration scheme.

The core computation in patch-based sampling can be formulated as a search for approximate nearest neighbors (ANN). We accelerate this search by combining optimized technique for General ANN search, a novel data structure called the quad tree pyramid for ANN search of images, and principal components analysis of the input sample texture. The patch-based sampling algorithm is easy to use and flexible. It is applicable to both unconstrained and constrained texture synthesis. Examples of constrained texture synthesis include hole filling and tileable texture synthesis. The patch-based sampling algorithm has an intuitive randomness parameter. The user can utilize this parameter to interactively control the randomness of the synthesized texture.

In the present study, Efros and Freeman developed a texture quilting algorithm. For unconstrained texture synthesis, texture quilting is very similar to patch-based sampling. There are several differences between our work and that of Efros and Freeman First, we accelerate patch-based sampling and demonstrate that it can run in real-time, whereas they do not explore the issue of speed. Second, we show how to solve constrained texture synthesis problems, which they do not address. Finally, patch-based sampling and Efros and Freeman] use different techniques to improve the transitions between texture patches. We apply feathering [Szeliski and Shum 1997] while Efros and Freeman use a minimum error boundary cut. Later, we compare these two boundary treatment techniques.

B. Patch Based Sampling

The patch-based sampling algorithm uses texture patches of the input sample texture I_{in} as the building blocks for constructing the synthesized texture I_{out}. In each step, we paste a patch B_k of the input sample texture I_{in} into the synthesized texture I_{out}. To avoid mismatching features across patch boundaries, we carefully select B_k based on the patches already pasted in I_{out}, fB_0: $B_{k°1}$g.

C. Sampling Strategy

Let I_{R1} and I_{R2} be two texture patches of the same shape and size. We say that I_{R1} and I_{R2} match if d (R_1, R_2) < ±, where d() represents the distance between two texture patches and ± is a prescribed constant. Assuming the Markov property, the patch-based sampling algorithm estimates the local conditional MRF (FRAME or Gibbs) density $p(I_R$ j $I_{@\ R}$) in a nonparametric form by an empirical histogram. Define the boundary zone @ R of a texture patch I_R as a band of width w_E along the boundary of R as shown in Figure 1b. When the texture on the boundary zone $I_{@\ R}$ is known, we would like to estimate the conditional probability distribution of the unknown texture patch I_R . Instead of constructing a model, we directly search the input sample texture I_{in} for all patches having the known $I_{@\ R}$ as their boundary zones. The results of the search form an empirical histogram *9* for the texture patch I_R . To synthesize I_R we just pick an element from *9* at random.

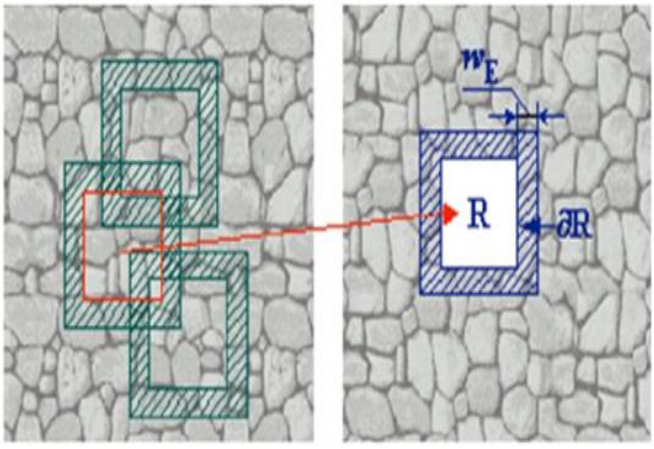

Fig1. A patch based sampling strategy. In the synthesized texture shown in (b) the hatched area is the boundary zone. In the input sample texture shown in (a), three patches have boundary zones matching the texture patch IR in (b) and the red patch is selected. (Source: H. Igehy -- 1997).

With patch-based sampling, the statistical constraint is implicit in the boundary zone *R*. On the down side, the nonparametric density estimation is subject to greater statistical fluctuations, because in a small sample texture I_{in} there may be only a few sites that satisfy the local statistical constraints.

1. Unconstrained Texture Synthesis

Now we use the patch-based sampling strategy to choose the texture patch B_k, the kth texture patch to be pasted into the output texture I_{out}. As Figure 2 shows, only part of the boundary zone of B_k overlaps the boundary zone of the already pasted patches fB_0 , B_k ◦1g in I_{out} . We say that two boundary zones match if they match in their overlapping region. In Figure Bk has a boundary zone E_{Bk} of width w_E . The already pasted patches in I_{out} also have a boundary zone E_{out}^k of width w_E . According to the patch-based sampling strategy, E_{Bk} should match E_{out}^k. For the randomness of the synthesized texture I_{out} , we form a set 9_B consisting of all texture patches of I_{in} whose boundary zones match E_{out}^k .

The patch-based sampling algorithm proceeds as follows:

1. Randomly choose a w_B £ w_B texture patch B_0 from the input sample texture I_{in}. Paste B_0 in the lower left corner of I_{out} . Set k D 1.
2. Form the set 9_B of all texture patches from I_{in} such that for each texture patch of 9_B, its boundary zone matches E_{out}^k .
3. If 9_B is empty, set 9_B D f$B_{\min}$g, where $B_{\min}$ is chosen such that its boundary zone is the closest to E_{out}^k .
4. Randomly select an element from 9_B as the kth texture patch B_k . Paste B_k onto the output texture I_{out}. Set k D k C 1.
5. Repeat steps 2, 3, and 4 until I_{out} is fully covered.
6. Perform blending in the boundary zones.

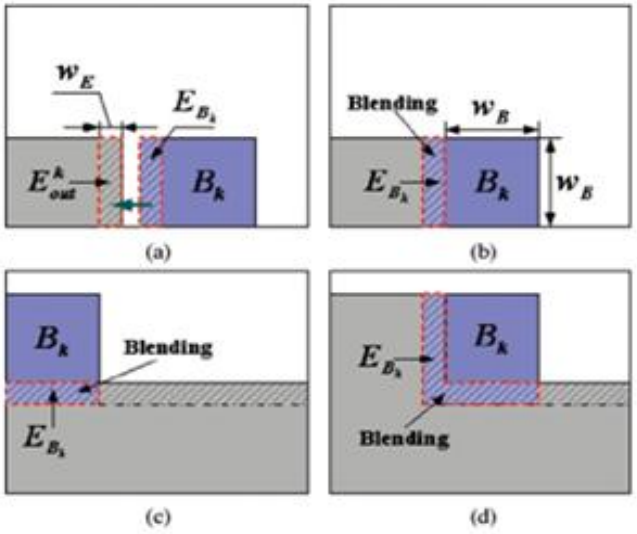

Fig2. Texture synthesis by patch-based sampling (source: D. Garber - 1981).

The grey area is already synthesized. The hatched area are the boundary zones.(a) The boundary zones E_{out}^k and E_{Bk} should match (b),(c) and (d) are three configurations for boundary zone matching. The overlapping boundary zones are blended together. The blending step uses feathering to provide a smooth transition between adjacent texture patches after I_{out} is fully covered with texture patches. Alternatively, it is possible to perform a feathering operation after each new texture patch is found.

D. Constrained Texture Synthesis

1. Hole Filling:

It is straightforward to extend the patch-based sampling algorithm to handle constrained texture synthesis. To better match the features across patch boundaries between the known texture around the hole and newly pasted texture patches, we fill the hole in spiral order, shown in the figure 3.

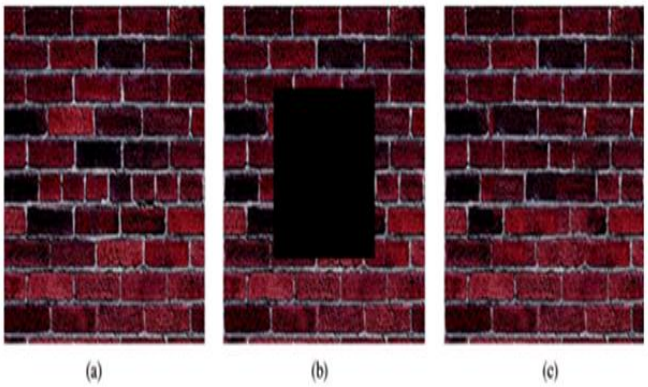

Fig3. Constrained texture synthesis: (a) 265£256 texture; (b) 128 £128 holes is created; (c) result of constrained synthesis.

2. Tileable Texture Synthesis:

This is another form of constrained texture syn-thesis. Figure 4 shows the boundary zones to be matched for the last patch in a row and for the patches of the last row. In the synthesized texture I_{out} , the pixel values in the boundary zone should be in the following equation.

$$E_{out}(x, y) \text{ D } I_{out}(x \bmod w_{out}, y \bmod h_{out}) \quad (1)$$

Where (x, y) is the location of the pixel in I_{out} . w_{out} and h_{out} are the width and height of the synthesized texture. This equation defines the pixels in the boundary zone E_{out} even if either $x > w_{out}$ or $y > h_{out}$ as shown in Figure. 4

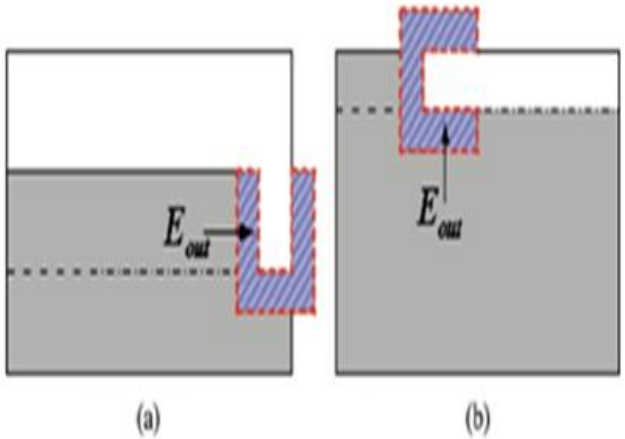

Fig4. Boundary zone matching for tileable texture synthesis. The grey area is the texture already synthesized. The hatched purple areas are the areas to be matched, (a) Boundary zone matching for the last patch in a row. (b) Boundary zone matching for the last row.

E. Review of Image Imapinting:

Inpainting, a set of techniques for making undetectable modifications to images, is as ancient as art itself. Applications of image inpainting range from the removal of an object from a scene to the retouching of a damaged painting or photograph. For photography and film, inpainting can be used to reverse deterioration (e.g., cracks in photographs, scratches and dust spots in film), or to add or remove elements (e.g., stamped dates, "red-eye," and, infamously, political enemies) from photographs [1]. In each case, the goal is to produce a modified image in which the inpainted region is merged into the image so seamlessly that a typical viewer is not aware that any modification has occurred.

The goals and applications of inpainting are numerous, from the restoration of damaged paintings and photographs to the removal/replacement of selected objects. We introduce a novel algorithm for digital inpainting of still images that attempts to replicate the basic techniques used by professional rest orators. After the user selects the regions to be restored, the algorithm automatically fills-in these regions with information surrounding them. The fill-in is done in such a way that isophote lines arriving at the regions boundaries are completed inside. In contrast with previous approaches, the technique here introduced does not require the user to specify where the novel information comes from. This is automatically done (and in a fast way), thereby allowing to simultaneously fill-in numerous regions containing completely different structures and surrounding backgrounds. In addition, no limitations are imposed on the topology of the region to be inpainted. Applications of this technique include the restoration of old photographs and damaged film; removal of superimposed text like dates, subtitles, or publicity; and the removal of entire objects from the image like microphones or wires in special effects.

Figure5 the work of our inpainting procedure starts with robust algorithm of color segmentation: mean shift segmentation. After we separate out structure information form image, the missing contour could be repaired by using Bézier curve. Finally, the exemplar-based image inpaintng method would be applied to recover all information of damaged area from other source area. The basic condition under the both side of the inpainting domain has the same amount of *n* contour lines as figure shows. And then to extend the work for salving the situation of *(n+1)* contour lines to *(n)* contour lines which in both side respectively as figure 6 shows. In this paper, we first construct the contour lines of image requiring inpainting, and then estimate the slope of contour line. Use them to estimate how they should extend and where two extended contour lines will meet.

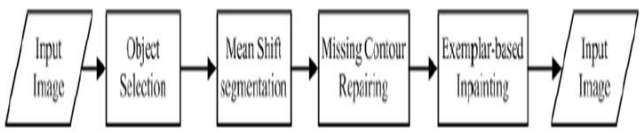

Fig5. System flow chart.

II. METHODOLOGY

A. Essential Techniques

In order to provide a flawless result of inpainting, there are several kinds of techniques such as mean shift segmentation, exemplar-based image inpainting and Bézier curves can be adopted.

B. Colour Segmentation by Mean Shift Segmentation

For inpainting, the contour lines have to be constructed for the image required. In the field of low level computer vision, many color segmentation and edge detection method had been proposed. From the well-known Sobel edge detection to canny edge detection, most of them are too sensitive to texture of cartoons. Mean-Shifts based color segmentation has to be selected to minimize the colors and textures and to generate the contour lines using edge detection.

1. Exempler Based Image Impainting

The contribution from scientists introduced two important fundamental techniques of image inpainting: priority map (i.e., P(p)) and conference term (i.e., C(p)). The priority map, P(p) = C(p)*D(p), uses a data term (i.e., D(p)) in additional to using the confidence term. The data term is based on the product of isophote and a unit vector orthogonal to the front of an inpainted area. The confidence term favors out-pointing regions (e.g., sharp bands) in the inpainted area. Thus, smooth or near circular inpainting area results. On the other hand, the data term gives a high priority to inpainted area which has a potential to copy structural information from the source area. The data term and the confidence term together results in a balance to select the best patch to be inpainted. When the inpainting algorithm selects a patch from the source area, sum of squared differences (SSD) using the CIE Lab color space is implemented.

In addition to using the similar approach, we introduce a new matching strategy, based on Edge Crispening and a new concept of patch template, to make sure that the best structural similarity is achieved. Let ***I*** be the original image (or a frame in a video) which includes a target area, denoted by Ω, to be inpainted and a source area, denoted by Φ, where patches are searched and used. Hence, ***I*** = Φ U Ω. A simple region segmentation algorithm based on the CIE Lab color space is used to convert ***I*** to ***I'***. Let *pi* and *pj* be pixels and *si* and *sj* be segments.

$$\forall p_i \in I,\ \forall p_j \in I,\ p_i \neq p_j \text{ and } p_i \text{ is adjacent to } p_j,$$

$$SSD_{CIE\ Lab}(p_i, p_j) < \delta_c \Rightarrow \text{make } p_i, p_j \text{ in the same segment}$$

$$\forall s_i \in I,\ \forall s_j \in I,\ s_i \text{ is adjacent to } s_j,$$

$$pn(s_i) - pn(s_j) < \delta_{pn} \Rightarrow \text{make } s_i, s_j \text{ in the same segment} \quad (1)$$

Where pn(s) computes the number of pixels in a segment. And, SSD CIE Lab (pi , pj) calculates the sum of squared differences using the CIE Lab color space. According to experiments, If δc is less than 3, the segmentation is hardly perceptible. If it is greater than 6, the result is not useful. We set the threshold δc to 3 simply to keep more segments for edge analysis. The second threshold *δpn* is different according to video. We manually adjust this threshold for

each video. Thus, ***I'*** is the result of color segmentation by using mean shift segmentation. The Edge Crispening algorithm is then used to convert ***I'*** to a binary image ***BI***, which represents an edge map (i.e., 0 for background and 1 for edge). Let Φε ∈ ***BI*** be the area corresponding to Φ ∈ ***I***. Φε is the edge map of the source region.

Let δΩ be a front contour on Ω and adjacent to Φ. We compute the initial Confidence Value of each pixel in ***I***, which is due to C(p) = 1 *iff* p ∈ Φ and C(p) = 0 *iff* p ∈ Ω. Let Ψp be a patch centered at pixel p ∈ δΩ. We also use the same concept to compute the confidence term C(p):

$$\forall\, p \in \delta\Omega,\ C(p) = \left(\sum_{q \in (\Psi_p \cap \Phi)} C(q)\right) / |\Psi_p| \qquad (2)$$

Where |Ψp| is the area of Ψp. essentially, the confidence term computes the percentage of useful pixels in a patch Ψp. Useful pixels mean the coverage of source pixels in Φ.

2. Bezier Curves

Bézier curve is an efficient method for designing polynomial curves when you want to control their shape, outline or pattern module with an easy way. Many CAD works using Bézier curves to design their shape, outline or pattern module. The appearance of these objects in the real world is much similar to Bézier curve. When we want to reconstruct the contour lines, Bézier curve is the useful method. To describe how the path of the Bézier curve goes which depends on the given control points.

Given the called control points ***PCC***= {***P0***, ***P1***, ***Pi***..., ***Pn*** }, $0 \le i \le n$ for the Bézier curve. The general form of Bézier curve of degree *n* are

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} P_i (1-t)^{n-i} t^i$$

$$= P_0 (1-t)^n + \binom{n}{1} P_1 (1-t)^{n-1} t + \ldots + P_n t^n \ ,$$

$$t \in [0,1] \qquad \textbf{(3)}$$

Where t is a variable of time.

C. Region-Filling Algorithm

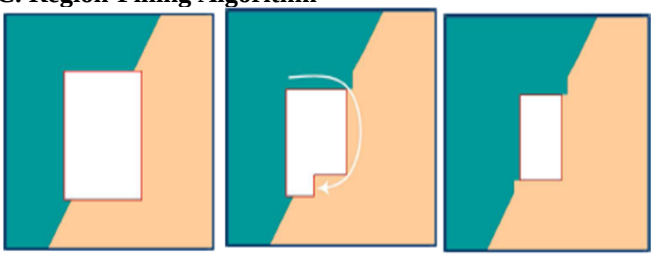

Fig6. The importance of the filling order in patch filling.(a) The target region is shown in white.(b) Part of the outmost layer in the target wise order.(c) At a further stage of the filling process,the remainder of the outmost layer has been filled by this clock wise onion – peel filling.The concentric layer filling has produced artefacts in the reconstruction of the diagonal image edge.A filling algorithm whose priority is guided by the image edges would fix this problem.(source - M. Bertalmio - 2001).

First, given an input image, the user selects a target region, Ω, to be removed and filled. The source region, Φ, may be defined as the entire image minus the target region $(\Phi = \mathcal{I} - \Omega)$, as a dilated band around the target region, or it may be manually specified by the user. .

1. The importance of the filling order in patch-based filling.

(a) The target region is shown in white. (b) Part of the outmost layer in the target region has been synthesized by an onion-peel algorithm proceeding in clockwise order. (c) At a further stage of the filling process, the remainder of the outmost layer has been filled by this clock-wise onion-peel filling. The concentric-layer filling has produced artifacts in the reconstruction of the diagonal image edge. A filling algorithm whose priority is guided by the image edges would fix this problem. Next, as with all exemplar-based texture synthesis, the size of the template window Ψ must be specified. We provide a default window size of 9 X 9 pixels, but in practice require the user to set it to be slightly larger than the largest distinguishable texture element, or "texel", in the source region. Once these parameters are determined, the region-filling proceeds automatically. In our algorithm, each pixel maintains a colour value (or "empty", if the pixel is unfilled) and a confidence value, which reflects our confidence in the pixel value, and which is frozen once a pixel has been filled. During the course of the algorithm, patches along the fill front are also given a temporary priority value, which determines the order in which they are filled. Then, our algorithm iterates the following three steps until all pixels have been filled:

2. Computing patch priorities:

Our algorithm performs the synthesis task through a best-first filling strategy that depends entirely on the priority values that are assigned to each patch on the fill front. The priority computation is biased toward those patches which: (i) are on the continuation of strong edges and (ii) are surrounded by high-confidence pixels. Given a patch ^{a}p centered at the point p for some $p \in \delta\Omega$, we define its priority P(p) as the product of two terms (fig 7):

$$P(p) = C(p)D(p) \qquad (4)$$

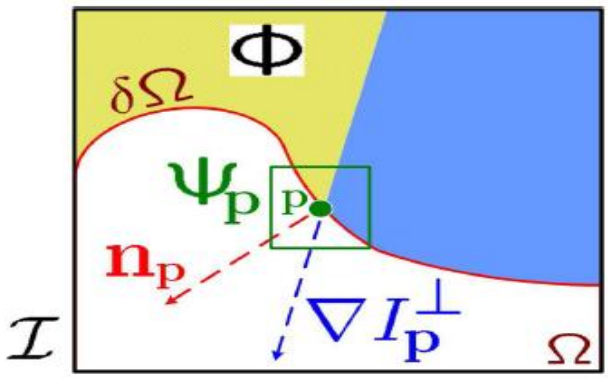

Fig7.

Notation diagram. Given the patch $\Psi_{\mathbf{p}}, \mathbf{n_p}$ is the normal to the contour $\delta\Omega$ of the target region Ω and $\nabla I_{\mathbf{p}}^{\perp}$ is the isophote (direction and intensity) at point p. The entire image is denoted with I. We call C(p) the confidence term and D(p) the data term, and, they are defined as follows:

$$C(\mathbf{p}) = \frac{\sum_{\mathbf{q}\in\Psi_{\mathbf{p}}\cap(\mathcal{I}-\Omega)} C(\mathbf{q})}{|\Psi_{\mathbf{p}}|}, \qquad D(\mathbf{p}) = \frac{|\nabla I_{\mathbf{p}}^{\perp}\cdot \mathbf{n_p}|}{\alpha} \tag{5}$$

Where $|\Psi_{\mathbf{p}}|$ is the area of $\Psi_{\mathbf{p}}$, α is a normalization factor (e.g., α = 255 for a typical grey-level image), $\mathbf{n_r}$ is a unit vector orthogonal to the front $\delta\Omega$ in the point p and $\perp$ denotes the orthogonal operator. The priority P(p) is computed for every border patch, with distinct patches for each pixel on the boundary of the target region. During initialization, the function C (p) is set to C (p) = 0 $\forall \mathbf{p} \in \Omega$, and C(p) = 1 $\forall \mathbf{p} \in \mathcal{I} - \Omega$. The confidence term C(p) may be thought of as a measure of the amount of reliable information surrounding the pixel p. The intention is to fill first those patches which have more of their pixels already filled, with additional preference given to pixels that were filled early on (or that were never part of the target region).

As it will be illustrated, this automatically incorporates preference towards certain shapes of the fill front. For example, patches that include corners and thin tendrils of the target region will tend to be filled first, as they are surrounded by more pixels from the original image. These patches provide more reliable information against which to match. Conversely, patches at the tip of "peninsulas" of filled pixels jutting into the target region will tend to be set aside until more of the surrounding pixels are filled in. At a coarse level, the term C (p) of (1) approximately enforces the desirable concentric fill order. As filling proceeds, pixels in the outer layers of the target region will tend to be characterized by greater confidence values, and therefore be filled earlier; pixels in the centre of the target region will have lesser confidence values. The data term D (p) is a function of the strength of isophotes hitting the front ±- at each iteration. This term boosts the priority of a patch that an isophote "flows" into. This factor is of fundamental importance in our algorithm because it encourages linear structures to be synthesized first, and, therefore propagated securely into the target region. Broken lines tend to connect, thus realizing the "Connectivity Principle" of vision psychology.

3. Propagating texture and structure information

Once all priorities on the fill front have been computed, the patch $\Psi_{\hat{\mathbf{p}}}$ with highest priority is found. We then fill it with data extracted from the source region Φ. In traditional inpainting techniques, pixel-value information is propagated via diffusion. As noted previously, diffusion necessarily leads to image smoothing, which results in blurry fill-in, especially of large regions. On the contrary, we propagate image texture by direct sampling of the source region. Similar to [7], we search in the source region for that patch which is most similar to $\Psi_{\hat{\mathbf{p}}}$.[2] formally,

$$\Psi_{\hat{\mathbf{q}}} = \arg\min_{\Psi_{\mathbf{q}}\in\Phi} d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\mathbf{q}}) \tag{6}$$

Where the distance $d(\Psi_{\mathbf{a}}, \Psi_{\mathbf{b}})$ between two generic patches $\Psi_{\mathbf{a}}$ and $\Psi_{\mathbf{b}}$ is simply defined as the sum of squared differences (SSD) of the already filled pixels in the two patches. Pixel colours are represented in the CIE Lab colour space because of its property of perceptual uniformity3. Having found the source exemplar $\Psi_{\hat{\mathbf{q}}}$, the value of each pixel-to-be-filled, $\mathbf{p}'$ $|\mathbf{p}' \in \Psi_{\hat{\mathbf{p}}\cap\Omega}$, is copied from its corresponding position inside $\Psi_{\hat{\mathbf{q}}}$. This suffices to achieve the propagation of structure and texture information from the source Φ to the target region Ω, one patch at a time. In fact, we note that any further manipulation of the pixel values (e.g., adding noise, smoothing etc.) that does not explicitly depend upon statistics of the source region, is more likely to degrade visual similarity between the filled region and the source region, than to improve it.

4. Updating confidence values

After the patch $\Psi_{\hat{\mathbf{p}}}$ has been filled with new pixel values, the confidence C(p) is updated in the area delimited by $\Psi_{\hat{\mathbf{p}}}$ is given in eqn7

$$C(\mathbf{p}) = C(\hat{\mathbf{p}}) \quad \forall \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega. \tag{7}$$

This simple update rule allows us to measure the relative confidence of patches on the fill front, without image-specific parameters. As filling proceeds, confidence values decay, indicating that we are less sure of the colour values of pixels near the centre of the target region. A pseudo-code description of the algorithmic steps is shown in table I. The superscript t indicates the current iteration.

5. Some properties of our region-filling algorithm:

The effect of the confidence term is that of smoothing the contour of the target region by removing sharp appendices and making the target contour close to circular (fig 8 a, b). Also, it can be noticed that inwards-pointing appendices are discouraged by the confidence term (red corresponds to low priority pixels). Unlike previous approaches, the presence of the data term in the priority function.

1. Tends to favour inwards-growing appendices in the places where structures hit the contour (green pixels), thus achieving the desired structure propagation.
2. Valid patches must be entirely contained in Φ.
3. Euclidean distances in Lab colour space are more meaningful than in RGB space.

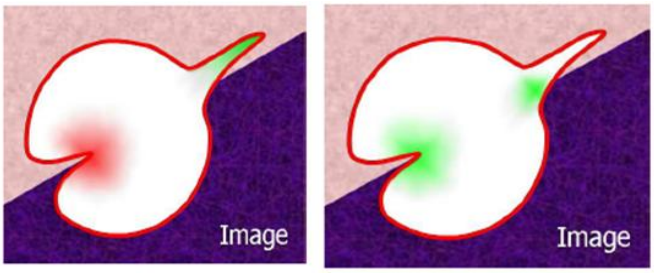

Fig8(a) Fig8 (b)

Fig8. Effects of data and confidence terms: 8(a) the confidence term assigns high filling priority to out-

pointing appendices (in green) and low priority to in-pointing ones (in red), thus trying to achieve a smooth and roughly circular target boundary. 8(b) the data term gives high priority to pixels on the continuation of image structures (in green) and has the effect of favoring in-pointing appendices in the direction of incoming structures. The combination of the two terms in produces the desired organic balance between the two effects, where the inwards growth of image structures is enforced with moderation.

6. Region filling algorithm

▪ Extract the manually selected initial front $\delta\Omega^0$.
▪ Repeat until done:
1a. Identify the fill front $\delta\Omega^t$. If $\Omega^t = \emptyset$, exit.
1b. Compute priorities P(p) $\forall \mathbf{p} \in \delta\Omega^t$.
2a. Find the patch $\Psi_{\hat{\mathbf{p}}}$ with the maximum priority, i.e., $\hat{\mathbf{p}} = \arg\max_{\mathbf{p}\in\delta\Omega^t} P(\mathbf{p})$.
2b. Find the exemplar $\Psi_{\hat{\mathbf{q}}} \in \Phi$ that minimizes $d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$.
2c. Copy image data from $\Psi_{\hat{\mathbf{q}}}$ to $\Psi_{\hat{\mathbf{p}}}\ \forall \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega$.
3. Update C(p) $\forall \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega$

But, as mentioned, the pixels of the target region in the proximity of those appendices are surrounded by little confidence (most neighbouring pixels are un-filled), and therefore, the "push" due to image edges is mitigated by the confidence term. As presented in the results section, this achieves a graceful and automatic balance of effects and an organic synthesis of the target region via the mechanism of a single priority computation for all patches on the fill front. Notice that (1) only dictates the order in which filling happens. The use of image patches for the actual filling achieves texture synthesis.

Furthermore, since the fill order of the target region is dictated solely by the priority function P(p), we avoid having to predefine an arbitrary fill order as done in existing patch-based approaches. Our fill order is function of image properties, resulting in an organic synthesis process that eliminates the risk of "broken-structure" artifacts Furthermore, since the gradient-based guidance tends to propagate strong edges, blocky and misalignment artifacts are reduced (though not completely eliminated), without a patch-cutting (quilting) step or a blur-inducing blending step. It must be stressed that our algorithm does not use explicit nor implicit segmentation at any stage.

7. Implementation details:

In our implementation the contour $\delta\Omega$ of the target region is modeled as a dense list of image point locations. These points are interactively selected by the user via a simple drawing interface. Given a point $\mathbf{p} \in \delta\Omega$, the normal direction $\mathbf{n}_\mathbf{p}$ is computed as follows: i) the positions of the "control" points of $\delta\Omega$ are filtered via a bi-dimensional Gaussian kernel and, ii) $\mathbf{n}_\mathbf{p}$ is estimated as the unit vector orthogonal to the line through the preceding and the successive points in the list. Alternative implementation may make use of curve model fitting. The gradient $\nabla I_\mathbf{p}$ is computed as the maximum value of the image gradient in $\Psi_\mathbf{p} \cap I$. Robust filtering techniques may also be employed here. Finally, pixels are classified as belonging to the target region Ω, the source region Φ or the remainder of the image by assigning different values to their alpha component. The image alpha channel is, therefore, updated (locally) at each iteration of the filling algorithm.

D. FLOW CHART

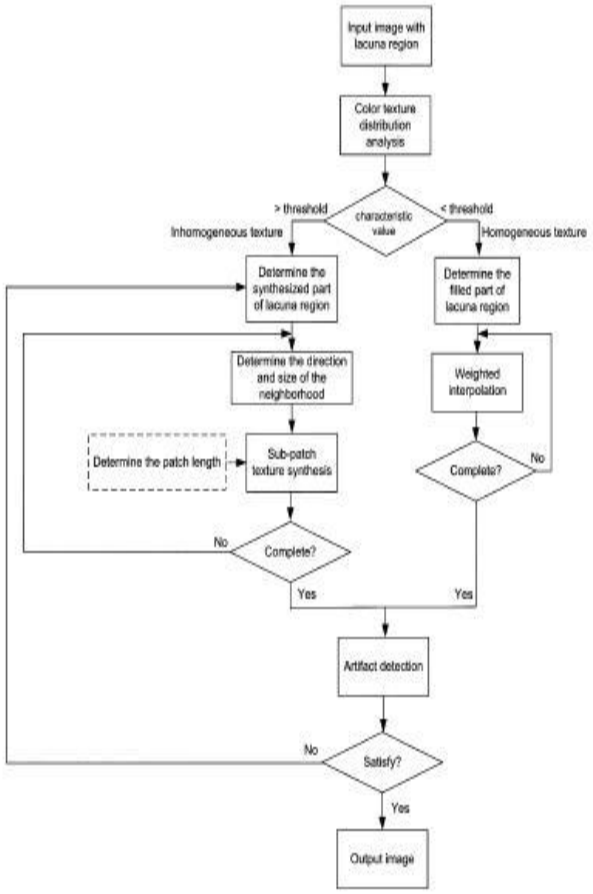

Fig9. Flow chart.

E. ALGORITHM

At first, color texture distribution analysis is performed on the Lacuna region to determine which method is to be used in the Lacuna region. After the first pass of filling the Lacuna region, the result with the filled Lacuna region is generated. Optionally, if the result is not satisfactory, the artifact detection mechanism will detect the artifact blocks in the filled region and color the artifact blocks in white. After the

second pass of region filling, these artifact blocks will be resynthesized by a subpatch texture synthesis technique to generate the final output image (fig 10).

Step	Description
1.	input an image with lacuna region (R_s , $R_l \neq \phi$)
2.	**foreach** pixel of lacuna region R_l
3.	**do** *Color Texture Distribution Analysis*
4.	**foreach** row of pixels of R_l
5.	**if** one of the values of *Color Texture Distribution Analysis* >Thre
6.	**then**
7.	**do** *Sub-patch Texture Synthesis Technique*
8.	**else**
9.	**do** *Weighted Interpolation Method*
10.	**until done**
11.	**if satisfied**
12.	**then**
13.	output an image without lacuna region (R_s , $R_l = \phi$)
14.	**else**
15.	**do** *Artifact Detection Mechanism*
16.	output an image without lacuna region (R_s , $R_l" \neq \phi$)
17.	**foreach** row of pixels of new lacuna region $R_l"$
18.	**do** *Sub-patch Texture Synthesis Technique*
19.	**until done**
20.	output an image without lacuna region (R_s , $R_l" = \phi$)

Fig10. Flow of Algorithm

III. RESULTS

Here we apply our algorithm to a variety of images, ranging from purely synthetic images to full colour photographs that include complex textures (fig 9a,b,c and 10a,b,c). Where possible, we make side-by-side comparisons to previously proposed methods. In other cases, we hope the original source of our test images (many are taken from previous literature on inpainting and texture synthesis) and compare these results with the results of earlier work. In all of the experiments, the patch size was set to be greater than the largest texel or the thickest structure (*e.g.*, edges) in the source region. Furthermore, unless otherwise stated the source region has been set to be *I*-.Ω All experiments were run on a 2.5GHz Pentium IV with 1GB of RAM.

A. Synthetic Image

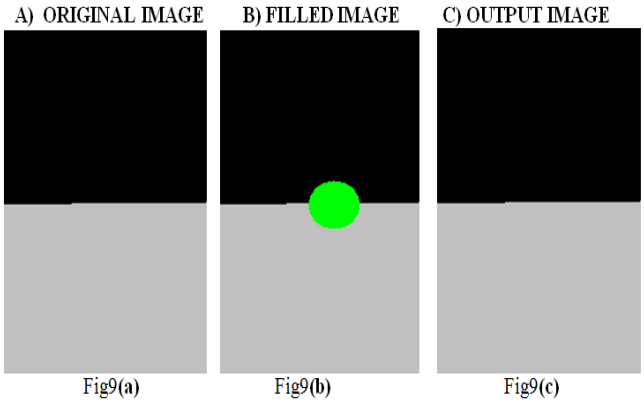

Fig9.

Case study 1:

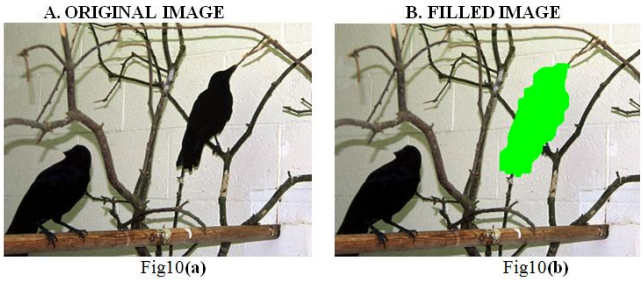

Fig10.

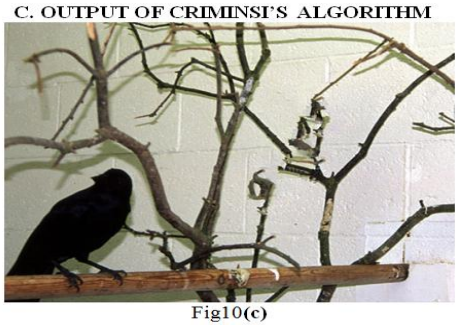

Fig10.

Case study 2:

Fig11.

IV. CONCLUSION AND FUTURE WORK

A novel algorithm has been proposed in this study for the replacement of objects on digital photographs. The result is an image in which the selected object has been replaced by a visually plausible background that mimics the appearance of the source region. Our approach employs an exemplar-based texture synthesis technique modulated by a unified scheme for determining the fill order of the target region. Pixels maintain a confidence value, which together with image isophotes, influence their fill priority. The technique is capable of propagating both linear structure and two-dimensional texture into the target region with a single, simple algorithm. Comparative experiments show that a simple selection of the fill order is necessary and sufficient to

handle this task. Our method performs at least as well as previous techniques designed for the restoration of small scratches, and, in instances in which larger objects are removed, it dramatically outperforms earlier work in terms of both perceptual quality and computational efficiency. Moreover, robustness towards changes in shape and topology of the target region has been demonstrated, together with other advantageous properties such as

1. Preservation of edge sharpness
2. No dependency on image segmentation and
3. Balanced region filling to avoid over-shooting artifacts.

Also, patch-based filling helps to achieve the speed efficiency, accuracy in the synthesis of texture (less garbage growing) and accurate propagation of the linear structures.
Limitations of the proposed technique s are

1. The synthesis of regions for which similar patches do not exist does not produce reasonable results.
2. The algorithm is not designed to handle curved structures and
3. It does not handle depth ambiguities (i.e., what is in front of what in the occluded area?).

Currently, the investigation is being carried out for the possibility of constructing ground-truth data and to design the evaluation tests that would allow the users to quantify the performance of individual algorithm. This turns out to be a non-trivial task. Furthermore the study is to be carried out for the extensions of the current algorithm to handle accurate propagation of curved structures in still photographs as well as removing objects from video, which promise to impose an entirely new set of challenges.

V. ACKNOWLEDGEMENTS

The authors would like to thank the faculty and colleagues of Raghu Engineering College, Vishakhapatnam, TataPower Bangalore and R V College of Engineering, Bangalore for the encouragement and infrastructure provided.

VI. REFERENCES

[1] M. Ashikhmin. Synthesizing natural textures. In Proc. ACM Symposium on Interactive 3D Graphics, pages 217–226, Research Triangle Park, NC, March 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 Proc. Int. Conf. Computer Vision, pages I: 10–16, Vancouver, Canada, June 2001.
[3] M. Bertalmio, A.L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In Proc. Conf. Comp. Vision Pattern Rec., pages I:355–362, Hawai, December 2001.
[4] S. Rane, G. Sapiro, and M. Bertalmio. Structure and texture filling-in of missing image blocks in wireless transmission and compression applications. In IEEE. Trans. Image Processing, 2002. to appear.
[5] H. Igehy and L. Pereira. Image replacement through texture synthesis. In Proc. Int. Conf. Image Processing, pages III:186–190, 1997.
[6] J. Jia and C.-K. Tang. Image repairing: Robust image synthesis by adaptive nd tensor voting. In Proc. Conf. Comp. Vision Pattern Rec., Madison, WI, 2003.
[7] D. Garber. Computational Models for Texture Analysis and Texture Synthesis. PhD thesis, University of Southern California, USA, 1981.
[8] P. Harrison. A non-hierarchical procedure for re-synthesis of complex texture. In Proc. Int. Conf. Central Europe Comp. Graphics, Visua. And Comp. Vision, Plzen, Czech Republic, February 2001.