ISSN:0975-9646

Exemplar-Based Image Inpainting for Region Filling & Object Removal

Rashmi Bijwe, Dr. P. R. Deshmukh
Computer Science, Amravati University,Sipna COET

Abstract -Image inpainting means reconstructions of missing or damaged portions of images. There are various algorithms 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, and (ii) "inpainting" techniques for filling in *small* image gaps. This paper presents an efficient algorithm for object removing and reconstructing it.

Keywords: *image inpainting, image restoration*

1. INTRODUCTION

Reconstruction of missing or damaged portions of images is an ancient practice used extensively in artwork restoration. Also known as *inpainting* or *retouching*, this activity consists of filling in the missing areas or modifying the damaged ones in a non-detectable way by an observer not familiar with the original images [1]. Applications of image inpainting range from restoration of photographs, films and paintings, to removal of occlusions, such as text, subtitles, stamps and publicity from images. In addition, inpainting can also be used to produce special effects.

Digital inpainting, the technique of reconstructing small damaged portions of an image, has received considerable attention in recent years. Digital inpainting serves a wide range of applications, such as removing text and logos from still images or videos, reconstructing scans of deteriorated images by removing scratches or stains, or creating artistic effects. Most inpainting methods work as follows. First, the image regions to be inpainted are selected, usually manually.

Next, color information is propagated inward from the region boundaries, i.e., the known image information are used to fill in the missing areas. In order to produce a perceptually plausible reconstruction, an inpainting technique should attempt to continue the isophotes (lines of equal gray value) as smoothly as possible inside the reconstructed region.

2. LITERATURE REVIEW & RELATED WORK

In previous work, several researchers have considered texture synthesis as a way to fill large image regions with "pure" textures – repetitive two-dimensional textural patterns with moderate stochasticity. This is based on a large body of texture synthesis research, which seeks to replicate texture *ad infinitum*, given a small source sample of pure texture [1]. Of particular interest are *exemplar-based techniques* which cheaply and effectively generate new texture by sampling and copying colour values from the source.

As effective as these techniques are in replicating consistent texture, they have difficulty filling holes in photographs of real world scenes, which often consist of linear structures and *composite textures* – multiple textures interacting spatially.The main problem is that boundaries between image regions are a complex product of mutual influences between different textures. In constrast to the two-dimensional nature of pure textures, these boundaries form what might be considered more one-dimensional, or linear, image structures.

A number of algorithms specifically address the image filling issue for the task of image restoration, where speckles, scratches, and overlaid text are removed [2].These *image inpainting* techniques fill holes in images by propagating linear structures into the target region via diffusion. They are inspired by the partial differential equations of physical heat flow, and work convincingly as restoration algorithms. Their drawback is that the diffusion process introduces some blur, which becomes noticeable when filling larger regions.

The technique presented here combines the strengths of both approaches into a single, efficient algorithm. The algorithm presented in this paper builds on very recent research along similar lines. The work in [3] decomposes the original image into two components; one of which is processed by inpainting and the other by texture synthesis. The output image is the sum of the two processed components. This approach still remains limited to the removal of small image gaps, however,as the diffusion process continues to blur the filled region . The automatic switching between "pure texture-" and "pure structure-mode" of is also avoided.

Similar to [3] is the work , where the authors describe an algorithm that interleaves a smooth approximation with example-based detail synthesis for image completion. The algorithm in [3] is extremely slow and it may introduce blur artefacts. In this paper we present a simpler and faster region filling algorithm which does not suffer from blur artefacts.

One of the first attempts to use exemplar-based synthesis specifically for object removal was by Harrison [3].

Although the intuition is sound, strong linear structures were often overruled by nearby noise, minimizing the value of the extra computation. A related technique drove the fill order by the local shape of the target region, but did not seek to explicitly propagate linear structures [4].

3. KEY OBSERVATION

A. Exemplar-based synthesis suffices

The core of presented algorithm is an isophote-driven image sampling process. It is well-understood that exemplar-based approaches perform well for two-dimensional textures [1][7]. But, in addition to this exemplar-based texture synthesis is sufficient for propagating extended linear image structures, as well; *i.e.*, a separate synthesis mechanism is not required for handling isophotes.

Now, the focus is on a single iteration of the algorithm to show how structure and texture are adequately handled by exemplar based synthesis. Suppose that the square template centred at the point, is to be filled. The best-match sample from the source region comes from the patch which is most similar to those parts that are already filled. In this paper the focus is on a patch-based filling approach (as opposed to pixel-based ones as in [7]) because, this improves execution speed. Also, the patch based filling improves the accuracy of the propagated structures.

B. Filling order is critical

This section shows that the quality of the output image synthesis is highly influenced by the order in which the filling process proceeds. Furthermore, There is a list of number of desired properties of the "ideal" filling algorithm.

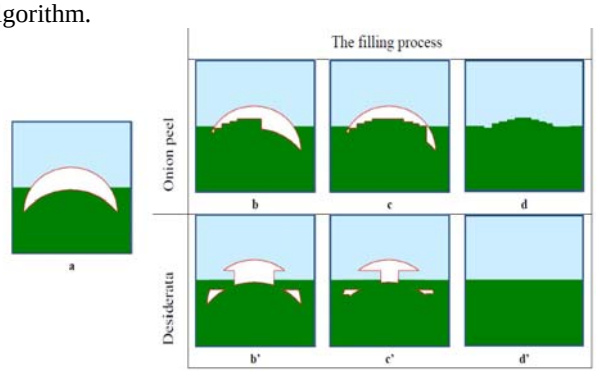

Fig. 1. The importance of the filling order when dealing with concave target regions.

A comparison between the standard concentric layer filling (onion-peel) and the desired filling behaviour is illustrated in fig. 1. Figures 1b,c,d show the progressive filling of a *concave* target region via an anti-clockwise onion-peel strategy. As it can be observed, this ordering of the filled patches produces the horizontal boundary between the background image regions to be unexpectedly reconstructed as a curve.

A better filling algorithm would be one that gives higher priority of synthesis to those regions of the target area which lie on the continuation of image structures, as shown in figs. 1b',c',d'.Together with the property of correct propagation of linear structures, the latter algorithm would also be more robust towards variations in the shape of the target regions.

A concentric-layer ordering, coupled with a patch-based filling may produce further artefacts. Therefore, filling order is crucial to non-parametric texture synthesis [1].

The desired property of a good filling algorithm is that of avoiding "over-shooting" artefacts that occur when image edges are allowed to grow indefinitely. The goal here is finding a good balance between the propagation of structured regions and that of textured regions. The filling algorithm overcomes the issues that characterize the traditional concentric-layers filling approach and achieves the desired properties of: (i) correct propagation of linear structures, (ii) robustness to changes in shape of the target region, (iii) balanced simultaneous structure and texture propagation, all in a single, efficient algorithm.

4. REGION-FILLING ALGORITHM

First, given an input image, the user selects a target region, to be removed and filled. The source region, may be defined as the entire image minus the target region, as a dilated band around the target region, or it may be manually specified by the user. Next, as with all exemplar-based texture synthesis, the size of the template window must be specified. The default window size of 9 * 9 pixels is provided, 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.

a b c

Fig.2. The importance of the filling order in patch-based filling.

In this algorithm, each pixel maintains a *colour* value (or "empty", if the pixel is unfilled) and a *confidence* value, which reflects the 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, the algorithm iterates the following three steps until all pixels have been filled:

1. Computing patch priorities.
The 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.
Consider a patch centred at the point p for some .its priority is define *P*(p) as the product of two terms:

$$P(p) = C(p)D(p) \quad (1)$$

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).
The 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) 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 this algorithm because it encourages linear structures to be synthesized first, and, therefore propagated securely into the target region.

2. Propagating texture and structure information.
Once all priorities on the fill front have been computed, the patch with highest priority is found. Then fill it with data extracted from the source region. 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, image texture is propagated by direct sampling of the source region.The search in the source region for that patch. Having found the source *exemplar*, the value of each pixel-to-be-filled is copied from its corresponding position inside.

3. Updating confidence values.
After the patch has been filled with new pixel values, the confidence is updated in the area .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.

Some properties of region-filling algorithm.

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 ,thus achieving the desired structure propagation 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 [7].
The steps in Region filling algorithm are shown below.

- Extract the manually selected initial front.
- Repeat until done:
  1a. Identify the fill front, exit.
  1b. Compute priorities.
  2a. Find the patch with the maximum priority,
  2b. Find the exemplar that minimizes distance.
  2c. Copy image data.
  3. Update *C*(p).

Furthermore, since the fill order of the target region is dictated solely by the priority function, It is avoid having to predefine an arbitrary fill order as done in existing patch-based approaches [7]. The fill order is function of image properties, resulting in an organic synthesis process that eliminates the risk of "broken-structure" artefacts. Furthermore, since the gradient-based guidance tends to propagate strong edges, blocky and mis-alignment artefacts are reduced, without a patch-cutting or a blur-inducing blending. It is stressed that this algorithm does not use explicit nor implicit segmentation at any stage.

5. APPLICATION

- Image inpainting is used for film restoration in photography.
- It is used for removal of occlusions like cracks in photographs or scratches and dust spots in film.
- It is used for removing the stamped date from photographs and removing objects . It can also be used to remove logos in videos.
- It can also be used to produce special effects.

6. CONCLUSION

This paper presents an algorithm for removing *large* objects from 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. 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.

7. 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. R. Bornard, E. Lecan, L. Laborelli, and J-H. Chenot. Missing data correction in still images and image sequences. In *ACM Multimedia*, France, December 2002.
3. A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *Proc. Conf. Comp. Vision Pattern Rec.*, Madison, WI, Jun 2003.
4. I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. In *ACM Trans. on Graphics (SIGGRAPH 2003 issue), 22(3)*, volume 22, pages 303–312, San Diego, US, 2003.
5. 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.
6. A. Criminisi, P. P´erez and K. Toyama, "Region Filling and Object Removal by Exemplar-Based Image Inpainting," *IEEE Trans on Image Processing, Vol. 13, No. 9, Sep 2004*
7. A. Efros andW.T. Freeman. Image quilting for texture synthesis and transfer.In *Proc. ACM Conf. Comp. Graphics (SIGGRAPH)*, pages 341–346,Eugene Fiume, August 2001.