Modified Image Exemplar-Based Inpainting

Mrs. Waykule J.M.,
Electronics, WCE Sangli, India

Abstract: This paper presents a way to modify the existing Exemplar-Based Image Inpainting. The increased processing time required for this algorithm will be essential to achieve perceptual difference in the quality of filling. The main focus is to better the priority function which will be reflected in the results in contrast to the unmodified algorithm. A new algorithm is planned for removing large objects from digital images. The challenge is to fill in the hole that is left behind in a visually believable way. In ancient times, this type of difficulty has been calculated by two classes of algorithms: (i) “texture synthesis” algorithms (ii) “inpainting” techniques. This paper presents a novel and efficient algorithm that combines the advantages of these two approaches but applied this same algorithm on modified image this gives the good result on one fourth less time to fill the target region.

Keywords: Image Inpainting, Texture Synthesis, Simultaneous Texture and Structure transmission, Exemplar-Based Image Inpainting

I. INTRODUCTION
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 by dividing the image and get good result in less time as compare to the without dividing the image.

Figure 1 show an example of this task, where the foreground lamp (manually selected as the target region) is automatically replaced by data sampled from the remainder of the image.

![Original Lamp Image](a)
![Mask image](b)
![Lamp successfully removed](c)
![Lamp successfully removed](d)

Fig.1 Removing Lamp objects from images. (a) Original Lamp Image (Size142x216). (b) Mask image (Total no. of pixels To Fill is 3047) (c) Lamp is successfully removed using Exemplar Based Inpainting.Required Execution time for this is 46.265000 Seconds. (d) Lamp is Successfully Removed using Divided Image in Exemplar Based Inpainting. Required Execution time for this is 10.891000 Seconds.

In Fig.1 (a) Shows Original Lamp Image, Our objective is to remove the lamp completely from this original image.(b) shows the manually selected this Lamp portion and he corresponding region filled in automatically shown in fig(c) using Exemplar Based Inpainting require four times more time as compare to Divided Image in same algorithm and both gives good result. Therefore Divided Image in Exemplar Based Inpainting is good here as compare to time and result. Notice that our algorithm succeeds in filling the target region without implicit or explicit segmentation.

II. PRESENT THEORY AND PRACTICES
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. The former work well for "textures"-repeating two-dimensional patterns with some stochastic; the latter focus on linear "structures" which can be thought of as one-dimensional patterns, such as lines and object contours.

III. KEY OBSERVATIONS
Exemplar-based synthesis suffices: The core of our algorithm is an isophote-driven image sampling process. It is well-understood that exemplar-based approaches perform well for two-dimensional textures [1], [11]. But, we note in addition that 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 the core of our algorithm is an isophote-driven image sampling process. It is well-understood that exemplar-based approaches perform well for two-dimensional textures [1], [11]. But, we note in addition that 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. Figure 3 illustrates this point. For ease of comparison, we adopt notation similar to that used in the inpainting literature. The region to be filled, i.e., the target region is indicated by Ω, and its contour is denoted δΩ. The contour evolves inward as the algorithm progresses, and so we also refer to it as the “fill front”. The source region, Φ which remains fixed throughout the algorithm, provides samples used in the filling process.

Fig. 3 Structure propagation by exemplar-based texture synthesis
(a) Original image, with the target region Ω, its contour δΩ, and the source region Φ clearly marked. (b) We want to synthesize the area delimited by the patch Ψp centered on the point p ∈ δΩ. (c) The most likely candidate matches for Ψp lie along the boundary between the two textures in the source region e.g. Ψp' and Ψq(Π+) (d) The best matching patch in the candidates set has been copied into the position occupied by Ψp, thus achieving partial filling of Ω. Notice that both texture and structure (the separating line) have been propagated inside the target region. The target region Ω

IV. REGION-FILLING ALGORITHM
First, given an input image, the user selects a target region, Ω, to be removed and filled. The source region, may be defined as the entire image minus the target region (Φ=I-Ω), it may be manually specified by the user. Algorithm iterates the following three steps until all pixels have been filled.

1) Computing Patch Priorities: The priority computation is biased toward those patches which: a) Are on the continuation of strong edges”) Are surrounded by high-confidence pixels. Given a patch Ψp centered at the point ‘p’. We define its priority as the product of two terms:

\[ P(P) = C(P)D(P) \]

\[ C(P) = \sum_{q \in \Psi_p} \left[ (q \in \Psi_p \cap (I-\Omega)) \cdot C(P) \right] \]

\[ D(P) = \frac{|VI_p \cdot \eta_p|}{\alpha} \]

Where, (a) \( \eta_p \) estimated as the unit vector orthogonal to Ψp the front δΩ... (b) VI_p is the isophote (direction and intensity) at point ‘p’ It computed as the maximum value of the image gradient in Ψp ∩ I (c) \( D(P) \) is the data term that is a function of the strength of isophotes hitting the front δΩ at each iteration.

2) Propagating Texture and Structure information: Propagate image texture by direct sampling of the source region, the distance \( d(\Psi_p, \Psi_q) \) between two generic patches Ψp and Ψq is simply defined as the sum of squared differences.

\[ \Psi_q = \arg \min_{\Psi_q \in \Phi} d(\Psi_p, \Psi_q) \]

3) Updating Confidence Values: After the patch Ψp has been filled with new pixel values, the confidence is updated in the area delimited by Ψp

\[ C = C(\Psi_p) \]

\[ V(P, \Psi_p) \in (\Psi_p \cap T) \]

V. IMPLEMENTATION DETAILS
Choice of Language: A variable alternative to using C++ would have been to use MATLAB. MATLAB is a high level language and interactive environment that enables you to perform computationally intensive task faster than with traditional programming languages such as
C,C++. Using MATLAB was considered due to the fact that its performance is better than C++ and contains a large number of built in image processing functions and also Image Library. The main reason of chosen was due to the fact that I have no prior experience of working with MATLAB Processing Steps
1. The program will take in two input images: the original image and the Photoshop mask that would mask out the object.
2. After read in the mask, I marked the target region that will be filled. Also; the contour (the boundary between the gap and the surrounding area) is defined as a collection of pixels that has a neighbouring pixel in the target region. I maintain a list of contour points in the form of a vector. For each pixel in the contour, I built a patch with a given size.
3. Then, I apply Criminisi’s algorithm to find out the target patch that has the highest priority to first fill according to filling order then next job is to find the best fit patch which is most similar patch to the highest priority patch. Originally this best fit patch find out by scanning whole image source region so this require large time to fill the target region so here I add the thing that is divide the image in four parts by dividing the rows by two and columns by two and minus the patch size because of patch based filling to care that the patch will not go to outside the image. Thus the image divides in to four parts. Therefore if we will find the best fit patch from whole image here we will take only the one part of the image for scanning or finding best fit patch. But how we can decide the part of image for finding the best fit patch, the solution is by checking the co-ordinates of highest priority pixel lies in which quadrant take that part for finding the best fit patch and thus the required time to fill the target region will become four times less with good result than the time required for scanning the whole image with same result. Thus, I performed a search to find a patch in the source area that is the most similar to the target patch or highest priority patch. In my implementation, I calculated the colour distance between the non-empty patch pixels at the same position. Sum of Square Difference (SSD) is used to calculate each colour channel’s difference, and then use SSD to sum up the overall colour distance between the pixels.
4. Once the best-fit patch has been found, I copy the colour values from the source patch to the target patch. A target patch contains portion of source region and portion of target region. Only those pixels in the target region will be filled.
5. After filling the patch, I renew the contour list. Those contour points that fall into the boundary of the target region will be removed from the list. At the same time, the pixels on the boundary of the target patch will be added to the contour list, if those pixels hadn’t been filled yet. This is illustrated by the Figure 4.4d.
6. I keep select patches whose centre point is on the contour to be filled. After the filling, I would renew the contour list. Eventually, the whole target region will be totally filled, and the contour list would be empty.
7. I keep select patches whose centre point is on the contour to be filled. After the filling, I would renew the contour list. Eventually, the whole target region will be totally filled, and the contour list would be empty.

VI. RESULT AND COMPARISONS
Here I apply this algorithm to a variety of images, ranging from purely synthetic images to full-colour photographs that include complex textures. Where possible, we make side-by-side comparisons to previously proposed methods. In other cases, I hope the reader will refer to 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 S-I-T. All experiments were run on a 2.5GHz Pentium IV with 1GB of RAM.
The experiment results show that the inpainted images are visually pleasant and computational efficiency is improved using Exemplar-Based Inpainting Method. It works well for all textured and structured Images and large objects removal.
In this project, I have also tested by dividing the image in four parts for same algorithm i.e. Exemplar-Based inpainting and check and compare the result of this with not divided the image in Exemplar-Based inpainting .Finally we compare speed and accuracy of a picture using Exemplar-Based Inpainting and Divided image in Exemplar-Based Inpainting. Divided the image in Exemplar-Based Inpainting is a good in computation time and accuracy than Exemplar-Based Inpainting. But Divided the image in Exemplar-Based Inpainting is not good for all type images. It completely fail on synthetic (paint) Images.

Synthetic image: We perform our first experiment on paint image, to show how the algorithm works on a structure-rich synthetic image.

Figure 6.1: Synthetic image (a) Original Synthetic image 307x206pix. (b) The target region to Fill (in red colour ). Total no of pixels to Fill are 531. (c) The result of region filling by our automatically Exemplar-based Image Inpainting algorithm .Time required to fill this target region is 5.68700sec and give the good result (d) The result of region filling by Divided Image in Exemplar-based Image Inpainting algorithm, Time required to fill this target region is 1.2500sec .Time required less but not
give the good result. Figure 6.1(c) shows how Exemplar-Based Inpainting algorithm achieves the best structural continuation in a simple, synthetic image. The sharp linear structures of the incomplete green triangle are grown into the target region as compared to Fig 6.1(d). Figure 6.1(d) shows result of modified Image Inpainting.

Fig 6.1(c) shows how Exemplar-Based Inpainting algorithm achieves the best structural continuation in a simple, synthetic image. The sharp linear structures of the incomplete green triangle are grown into the target region as compared to Fig 6.1(d). Figure 6.1(d) shows result of modified Image Inpainting.

Fig 6.3: Removing multiple objects from photographs. Fig 6.3 (a) Original Sea land Image (Size 207x279) with several objects (b) Object-Cut image (Total no. of pixels to fill are 5727). (c) Several objects are Successfully Removed using Example Based Inpainting. Required execution time for this is 178.453000 Seconds. (d) Several objects are Successfully Removed using Divided Image in Exemplar Based Inpainting. Required execution time for this is 46.406000 Seconds.

Fig 6.4: Removal of Scratches (a) Original damaged Scratch Image size (257x386). (b) Mask Image (Total no. of Pixels to fill are 1833). (c) Scratches are Successfully Removed using Exemplar Inpainting. Required execution time for this is 233.532000 Seconds. (d) Scratches are successfully removed using Divided Image in Exemplar Based Inpainting. Required execution time for this is 8.953000 Seconds. Patch sizes same for both cases are 11x11.

Fig 6.5: Hill images (a) Original Hill Image size 162x295. (b) Mask Image (Total no. of Pixels to fill are 1892). (c) Object is successfully removed using Exemplar Inpainting. Required execution time for this is 35.328000 Secs. (d) Scratches are successfully removed using Divided Image in Exemplar Based Inpainting. Required execution time for this is 8.953000 Secs. Patch sizes same for both cases are 11x11.

Fig 6.6: Boat image (a) Original Boat Image (257x342). (b) Object-Cut Image (Total no. of pixel to fill are 9502) (c) Result after removing Boat using Exemplar Based Image Inpainting with Patch size 11x11. Required time for fill is 431.921000 seconds. (d) Result after removing Boat using Divided Image Exemplar Based Image Inpainting with Patch size 11x11. Required time for fill is 108.672000 seconds.
Plot 1 shows that the Divided image in Exemplar-Based inpainting is good because it gives also a good result with four times less than the Exemplar-Based inpainting. When the target region to fill is more time required to fill this also more.

Table 3.3: Execution time for Exemplar-Based inpainting and Divided Image in Exemplar-Based inpainting

<table>
<thead>
<tr>
<th>Image</th>
<th>Size of Image</th>
<th>Total no. of pixel in image</th>
<th>Total no. of Pixels to Fill</th>
<th>Exemplar-Based inpainting, (sec)</th>
<th>Divided image in Exemplar-Based inpainting, (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lamp</td>
<td>142x216</td>
<td>30672</td>
<td>3047</td>
<td>46.2650</td>
<td>10.8910</td>
</tr>
<tr>
<td>Sea</td>
<td>207x279</td>
<td>57753</td>
<td>9502</td>
<td>178.4530</td>
<td>46.4060</td>
</tr>
<tr>
<td>Scratch</td>
<td>257x386</td>
<td>99202</td>
<td>1833</td>
<td>233.5320</td>
<td>61.4530</td>
</tr>
<tr>
<td>Hill</td>
<td>162x215</td>
<td>34830</td>
<td>1892</td>
<td>35.3280</td>
<td>8.9530</td>
</tr>
<tr>
<td>Boat</td>
<td>257x342</td>
<td>87894</td>
<td>9502</td>
<td>431.9210</td>
<td>108.6720</td>
</tr>
<tr>
<td>Nice</td>
<td>220x176</td>
<td>38720</td>
<td>1124</td>
<td>27.1410</td>
<td>7.2030</td>
</tr>
<tr>
<td>Text</td>
<td>199x257</td>
<td>51143</td>
<td>817</td>
<td>96.6410</td>
<td>32.5630</td>
</tr>
</tbody>
</table>

Figure 6.9: Removing Large objects from photographs.
I have tried various sizes of patches. I look the results on how well it blends with the surrounding area, and on if the shapes and structures are well preserved in the filled region. It seems that 7x7 has the best result. 9x9 blends well with the sea, but fails on the sky. The conclusion I drew from the patch size experience is that most of the texture elements is around 7 and 9, 11. The reason that other sizes fail might because of the bad samplings.

VII. CONCLUSION

This paper has presented a novel algorithm for removing large objects from digital photographs. The result is an image in which the selected object has been replaced by a visually believable background that mimics the look of the source region.

Our first 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. This first approach gives the good result both on Real images and on Synthetic Images.

Our second approach employs that divided the image in four parts for same algorithm i.e. Exemplar-Based Image Inpainting. This approach also gives a good result with less time as compare to our first approach (i.e. Exemplar-Based Inpainting) on Real images, but this second approach has been failed on Synthetic Images.

Advantages: a) Preservation of edge sharpness, No dependency on image segmentation. Balanced region filling to avoid over-shooting artifacts. Patch-based filling helps achieve: (i) speed efficiency, (ii) Accuracy in the synthesis of texture (iii) accurate propagation of linear structures. Another future work would be extending the algorithm, so that it could be used in object removal in video sequences. The current algorithm is still slow, and further improvements would need to increase the performance, so that this algorithm could be used to video applications.

REFERENCES

BIOGRAPHY

Assistant Prof. Mrs. Jyoti Waykule
completed M.Tech. (Electronics), I have published one international journal.