

# 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.

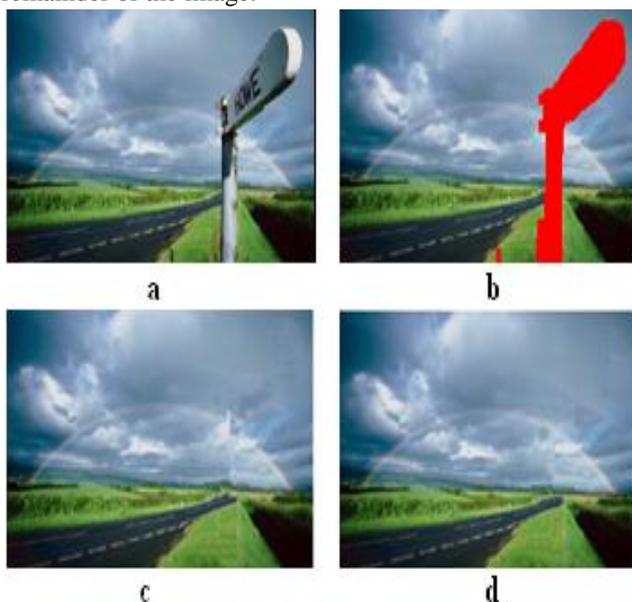


Fig.1 Removing Lamp objects from images. (a) Original Lamp Image (Size 142x216). (b) Mask image (Total no. of pixels To Fill is 3047) (c) Lamp is successfully removed using Exemplar Based Inpainting. Required Execution time for this is 46.265000 Seconds. (d) Lamp is Successfully Removed using Divided 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

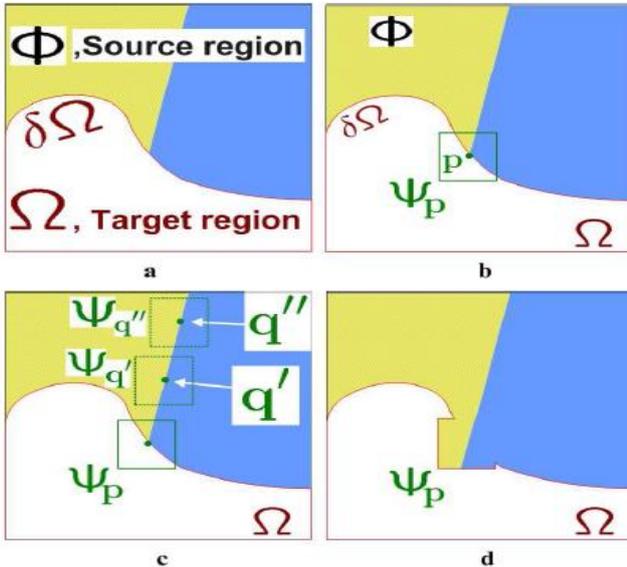
In the past, this problem has been addressed by two classes of algorithms: (i) "texture synthesis" algorithms for generating large image regions from sample textures, and (ii) "inpainting" techniques for filling in *small* image gaps. The former work well for "textures"-repeating two-dimensional patterns 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],[17]. 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  $\Omega$ , and its contour is denoted  $\delta\Omega$ . The contour evolves inward as the algorithm progresses, and so we also refer to it as the “fill front”. The source region,  $\Phi$  which remains fixed throughout the algorithm, provides samples used in the filling process.



has, now, shrunk and its front  $\delta\Omega$  has assumed a different shape. The user will be asked to select a target region,  $\Omega$ , manually. (a) The contour of the target region is denoted as  $\delta\Omega$ . (b) For every point  $p$  on the contour  $\delta\Omega$ , a patch  $\Psi_p$  is constructed, with  $p$  in the centre of the patch. A priority is calculated based on how much reliable information around the pixel, as well as the isophote at this point. (c) The patch with the highest priority would be the target to fill. A global search is performed on the whole image to find a patch,  $\Psi_q$  that has most similarities with  $\Psi_p$ . (d) The last step would be copy the pixels from  $\Psi_q$  to fill  $\Psi_p$ . With a new contour, the next round of finding the patch with the highest continues, until all the gaps are filled.

#### IV. REGION-FILLING ALGORITHM

First, given an input image, the user selects a target region,  $\Omega$ , to be removed and filled. The source region, may be defined as the entire image minus the target region ( $\Phi=I-\Omega$ ), 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  $\varphi_p$  centered at the point 'p'. We defines its priority  $P(P)$  as the product of two terms:

$$P(P) = C(P)D(P)$$

Fig.3 Structure propagation by exemplar-based texture synthesis . (a) Original image, with the target region  $\Omega$ , its contour  $\delta\Omega$ , and the source region  $\Phi$  clearly marked. (b) We want to synthesize the area delimited by the patch  $\varphi_p$  centered on the point  $p \in \delta\Omega$ . (c) The most likely candidate matches for  $\varphi_p$  lie along the boundary between the two textures in the source region e.g.  $\varphi_{p'}$  and  $\varphi_{1(P^{1*})}$  (d) The best matching patch in the candidates set has been copied into

the position occupied by  $\varphi_p$ , thus achieving partial filling of  $\Omega$ . Notice that both texture and structure (the separating line) have been propagated inside the target region. The target region  $\Omega$ .

$C(P)$  The confidence term that measure of the amount of reliable information surrounding the pixel 'p'

$$C(P) = \frac{\sum[(q \in \varphi_p) \cap (I-\Omega)]C(P)}{|\Psi_p|}$$

$D(P)$  The data term that is a function of the strength of isophotes hitting the front  $\delta\Omega$  at each iteration.

$$D(p) = \frac{|\nabla I_p^\perp \cdot \eta_p|}{\alpha}$$

Where, (a)  $n_p$  estimated as the unit vector orthogonal to  $\varphi_p$  the front  $\partial\Omega$ ... (b)  $\nabla_p^\perp$  Is the isophote (direction and intensity) at point 'p' It computed as the maximum value of the image gradient in  $\varphi_p \cap I$  (c)  $\alpha$ =Normalization factor (e.g. Gray level it is  $\alpha=255$ )

2) Propagating Texture and Structure information:

Propagate image texture by direct sampling of the source region, the distance  $d(\varphi_p, \varphi_q)$  between two generic patches  $\varphi_p$  and  $\varphi_q$  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  $\varphi_p$  has been filled with new pixel values, the confidence  $C(P)$  is updated in the area delimited by  $\varphi_p$

$$C(P) = C(P^*) \quad \nabla_i(P) \in (\varphi_{1(P^*)} \cap T)$$

#### 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.
5. 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.
6. 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 Figure4.4d
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.

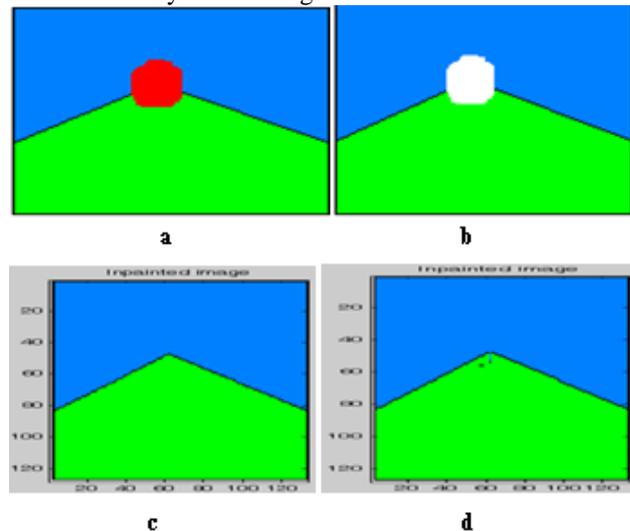


Figure 6.1: Synthetic image (a) Original Synthetic image 307x206pix. (b) The target region to Fill (in red colour), Total no of pixels to Fill are 531. (c) The result of region filling by our automatically Exemplar-based Image Inpainting algorithm, Time required to fill this target region is 5.68700sec 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 compare to fig 6.1(d). Figure 6.1(d) shows result of modified Image In Exemplar-Based Image Inpainting

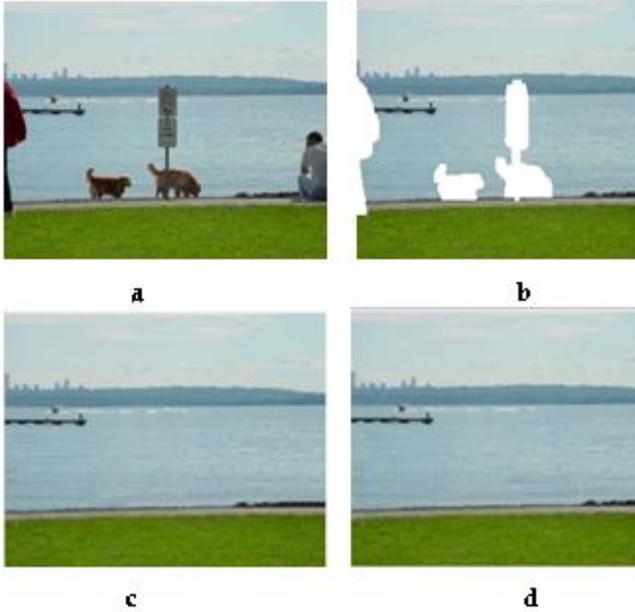
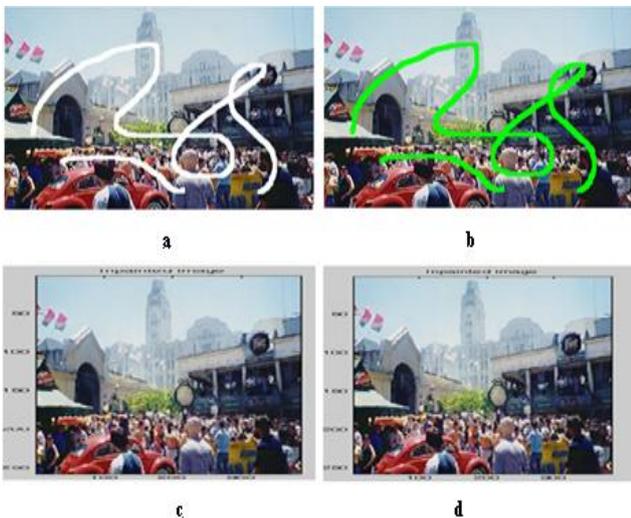


Fig 6.3: Removing multiple objects from photographs. Fig 6.3 (a) Original Sea land Image (Size 207x279) with Several objects (b) Object-Cut image (Total no. of pixels To Fill are 5727). (c) Several objects are Successfully Removed using Example Based Inpainting. Required execution time for this is 178.453000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

Fig6.4: 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 Based 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 61.453000 Seconds. Patch sizes same for both cases which are 11x11.

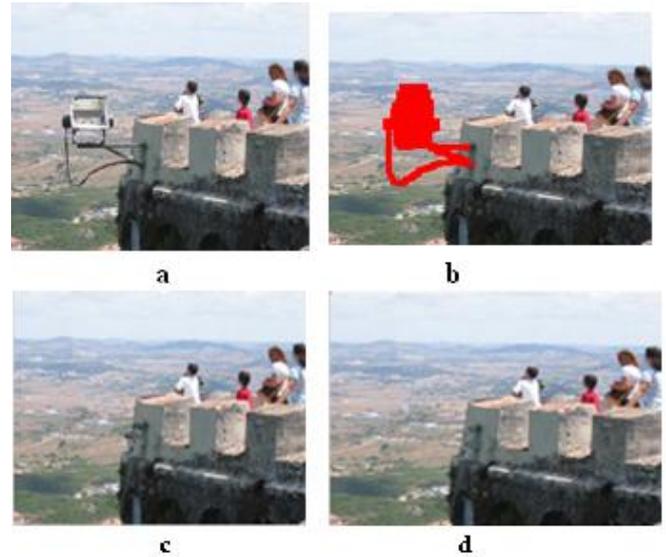


Fig 6.5: Hill images (a) Original Hill Image size 162x295. (b) Mask Image (Total no. of Pixels to fill are 1892). (c) Object is successfully removed using Exemplar Inpainting. Required execution time for this is 35.328000 Secs. (d) Scratches are successfully removed using Divided Image in Exemplar Based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

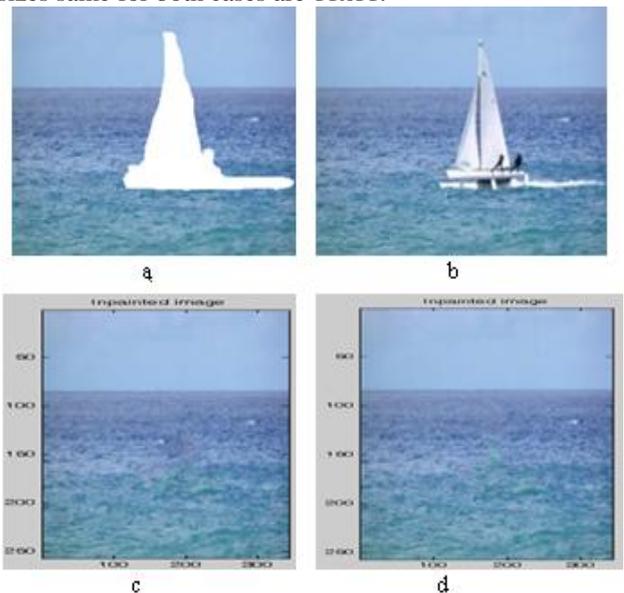


Fig 6.6: Boat image (a) Original Boat Image (257x342). (b) Object-Cut Image (Total no. of pixel to Fill are 9502) (c) Result after removing Boat using Exemplar Based Image Inpainting with Patch size 11x11. Required time for fill is 431.921000 seconds. (d) Result after removing Boat using Divided Image Exemplar Based Image Inpainting with Patch size 11x11. Required time for fill is 108.672000 seconds.

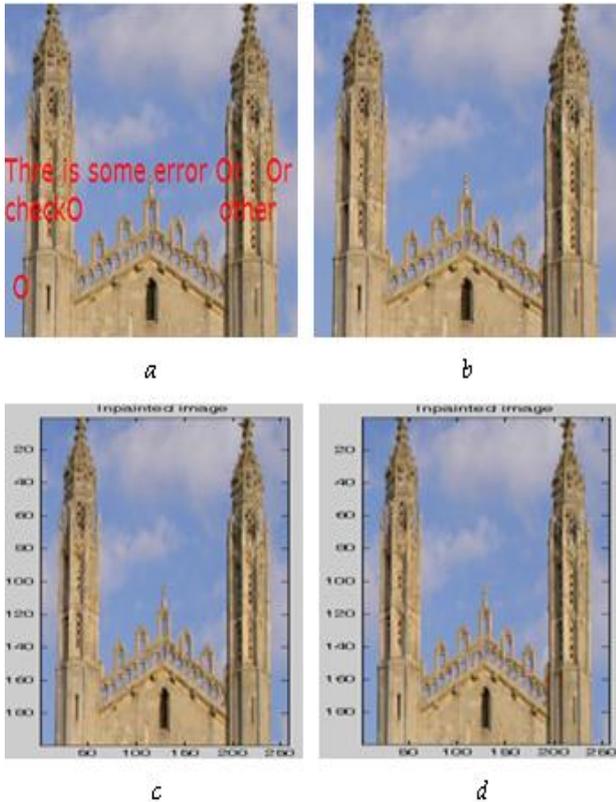
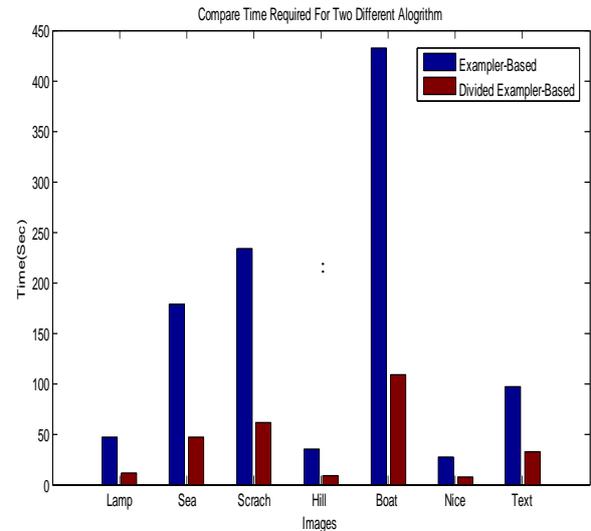


Fig 6.7: Text image (a) Original Text Image (199x257). (b) super- imposed Text Removed (Total no. of pixel to Fill are 817) (c) Result after removing Text using Exemplar Based Image Inpainting with Patch size 11x11. Required time for fill is 96.6410seconds. (d) Result after removing Text using Divided Image in Exemplar Based Image Inpainting with Patch size 11x11. Required time for fill is 32.5630 seconds. Thus by using this Divided Image in Exemplar Based Inpainting OR Modified Image Exemplar Based Inpainting on number of images and compare the result with Exemplar Based Image Inpainting and collect the result which show that,by using this method we required the less time to fill the target region of the image expect the paint images.

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

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

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.



Plot 1: Comparisons Of Time Required for Two different algorithms:

Comparisons of Time Required for Two different algorithm on same image one is Exemplar-Based Inpainting shown by blue bar and other is Divided image in Exemplar-Based inpainting shown by red bar. X-axis shows Lamp Image in different patch size and Y-axis shows Time (sec) required for different patch size.

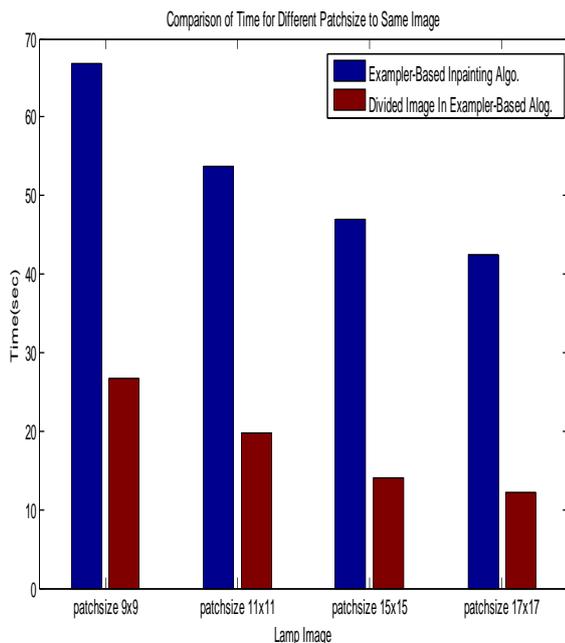


Figure 6.9: Removing Large objects from photographs.

(a)Original Lamp Image (Size 142x216) (b) Object-Cut image OR Mask Image (Total no. of pixels To Fill are 3047). (c) Lamp is not successfully removed using Exemplar Based Inpainting with patch size 9x9 .Required Execution time for this is 66.750 Sec. (d) (e) (f) Lamp is Successfully removed using Exemplar Based Inpainting with patch size 11x11,15x15 and 17x17 .Required Execution time for this is 53.765sec, 46.953sec,42.406 secs respectively (c') Lamp is not successfully removed using Divided Image In Exemplar Based Inpainting with patch size 9x9 .Required Execution time for this is 25.610Sec (d') (e') (f') Lamp is successfully removed using Divided Image In Exemplar Based Inpainting with patch size 11x11, 15x15and17x17 .Required Execution time for this is 19.790sec, 13.969sec,12.1720 sec respectively

Plot 2: Comparisons Of Time Required For Different Patch size to the Same Image: Comparisons Of Time Required For different patch size to the same image by using two algorithm one is Exemplar-Based Inpainting shown by blue bar and other is Divided image in Exemplar-Based inpainting shown by red bar. X-axis shows Lamp Image in different patch size and Y-axis shows Time (sec) required for different patch size.

In my results, I had experimented with a few parameter of the Criminisi algorithm. One important parameter of the algorithm is the size of the patch. With bigger patch size, the filling rate is high (which shows in plot2), thus the program runs faster. However, there's more important implication on choosing the right patch size.



Criminisi stated in his paper that the patch should be slightly larger than the largest distinguishable texture element. He gave 11x11 as default in his paper. As result shows, 9x9 is not the best choice in the sample image used in this report.

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

- [1] A.Criminisi P.Perez and K.Toyama. "Region Filling and Object Removal by Exemplar-Based Image inpainting" IEEE Transaction on image processing, vol 13, .SEP2004
- [2] P. Harrison. "A non-hierarchical procedure for re-synthesis of complex texture." In Proc. Int. Conf. Central Europe Comp. Graphics, February 2001
- [3] M.Bertalmio, L. Vese, G. Sapiro, and S. Osher. "Simultaneous structure and texture image inpainting." In Proc. Conf. Comp. Vision Pattern Rec., Madison, WI, 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] A.Efros and T. Leung. "Texture synthesis by non-parametric sampling." In Proc. Int. Conf. Computer Vision, pages 1033–1038, Kerkyra, Greece, September 1999
- [6] C.Ballester, V. Caselles, J. Verdera, M. Bertalmio, and G. Sapiro. "A variational model for filling-in gray level and colour images." In Proc. Int. Conf Computer June 2001
- [7] J.S.De Bonet. "Multiresolution sampling procedure for analysis and synthesis of texture images." In Proceedings of SIGGRAPH 1997
- [8] D.Garber. Computational Models for Texture Analysis and Texture Synthesis. PhD thesis, University of Southern California, 1981



- [9] M.M.Oliveira, B. Bowen, R. Mckenna, and Y. S. Chung, "Fast Digital Inpainting", Sep 2001
- [10] T.F.chan and J.Shen,"Mathematical model for local non-texture inpainting," 2001
- [11] R.J.Cant and C. S. Langensiepen, "A Multiscale Method for Automated Inpainting", ESM2003, 2003
- [12] J.Portilla and E.P.Simoncelli, "A parametric texture coefficients," International Journal of Computer vision, vol. 40, no. 1, 2000
- [13] D.J.Heeger and J.R. Bergen, "Pyramid-based texture analysis/ synthesis "in SIGGRAPH, 1995
- [14] J.Jia and C.-K. Tang, "Image repairing: Robust image synthesis by adaptive and tensor voting,"cvpr, vol. 01, p. 643, 2003
- [15] Websites: <http://www.mathworks.com/>, <http://www.ieeexplore.ieee.org/>
- [16] Reference Books:
  - a) Rafael C. Gonzalez, Richard E.Woods, "Digital Image Processing"
  - b) A. K. Jain." Fundamentals of Digital Image processing"

### **BIOGRAPHY**



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