ISSN: 2321-8169
1522 – 1530

Divided Exemplar-Based Image Inpainting

Assist. Prof. Mrs. Waykule Jyoti .M.,
Electronics, Sanjay Ghodawat Group Of Institutions, Atigre.
Kolapur,India.
(waykule.jm@sginstitute.in)

Abstract---This paper presents a method to apply 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. A method is planned for removing large objects from digital images by giving good result in four times less moment than the applying the Exemplar-Based Image Inpainting on whole image. The challenge is to fill in the hole that is left behind in a visually believable way with an efficient time.In ancient period; 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 divided image this gives the good result on one fourth less time to fill the target region.

Keywords:*ImageInpainting, Texture Synthesis,Simultaneous Texture and Structuretransmission,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.

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.

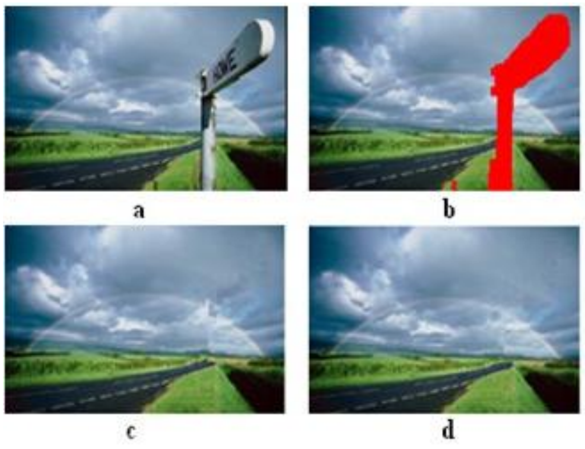


Fig.1 Removing Lamp objects from images.(a) Original Lamp Image (Size142x216). (b) Mask image (Total no. of pixels To Fill is 3047)(c) Lamp is successfully removed using Exemplar Based Inpainting.Required Execution time for this is 46.265000 Seconds. (d) Lamp is Successfully Removed using Divided Image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

In Fig.1 (a) ShowsOriginal 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. Outward from an initial seed.Effors and Leung [5] proposed a texture synthesis approach by non-parametric sampling. Texture synthesis grows a new image.

A number of algorithms specifically address the image filling issue for the task of image restoration, where speckles, scratches, and overlaid text are removed [3], [06].

The idea of digital image inpainting as first introduced by Bertalmo et al in their revolutionary paper “Image Inpainting” [06]. These image inpainting techniques fill holes in images by propagating linear structures (called isophotes in the inpainting literature) into the target region via diffusion. They are inspired by the partial differential equations of physical heat flow, their drawback is that the diffusion process introduces some blur, which becomes noticeable when filling larger regions.

The conventional schemes for inpainting are divided into two different types namely texture oriented and structure oriented. The texture-oriented scheme (usually known as the texture synthesis scheme) generates the target region with available sample textures from its surroundings. This approach is specifically useful for the images with large texture areas. On the other hand, the structure-oriented scheme obtains the missing regions via data fusion techniques, such as the bilinear interpolation. However, since most ordinary images are not composed of pure texture or pure structure, better performance is expected for those taking advantages of both schemes. Bertalimo [3] proposed to decompose each original image into two separate component images of different characteristics. One of them is processed by texture oriented scheme and the other by structure-oriented one. The two processed components are added back to reconstruct the missing regions. 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" is also avoided.

Similar to [3] is the work in [4], where the authors describe an algorithm that interleaves a smooth approximation with example-based detail synthesis for image completion. Like the work in [3] also the algorithm in [4] is extremely slow (as reported processing may take between 83 and 158 minutes on a 384x256 image) and it may introduce blur artifacts. The method presented in Fast Digital Image Inpainting [09] is specifically designed for inpainting relatively small regions. When focusing on small regions simpler models can be used. Instead of relying on any specific mathematical geometrical theories, this method takes into account a constraint of the sampling theorem.

One of the first attempts to use exemplar-based synthesis specifically for object removal was by Harrison [2]. A number of texture re-synthesis methods are described in the literature. One approach is based on searching for specific features in textures [07] [5] .In these methods, the input image is decomposed into a set of features. Statistics about these features are collected, and used to synthesize a new image. One problem with these methods is that they can only recognize a set of features which have been specied in advance. While the results can be impressive, it is difficult to devise a generic feature set that can be used to describe all textures. Another approach to the texture re-synthesis problem is to analyses and reproduce interactions between individual pixels [08] .Methods based on this approach define a function describing a pixel in terms of its neighbors and a random element. This function is used to generate a new image, pixel by pixel.

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 isophotesthe 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 theinpaintingjournalism. 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.

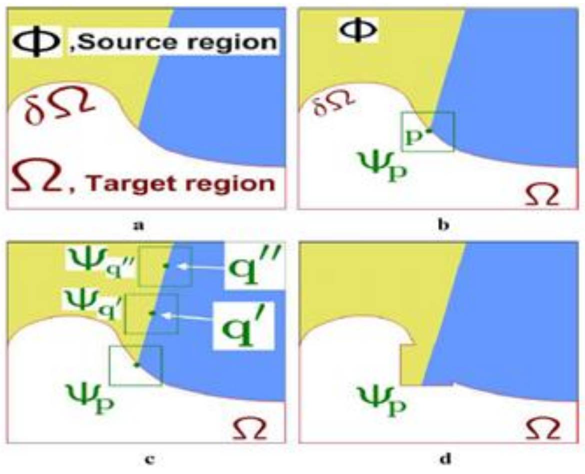


Fig.3 Structure propagation by exemplar-based texture synthesis

. (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 φ_p centered on the point p ε ∂Ω. (c) The most likely candidate matches for φ_p lie along the boundary between the two textures in the source region e.g. $\varphi_{p'}$ and $\varphi_{\downarrow}(P^{\uparrow''})$ (d) The best matching patch in the candidates set has been copied into the position occupied by ψ_P , 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 has, now, shrunk and its front δΩ.has assumed a different shape. The user will be asked to select a target region, Ω, manually. (a) The contour of the target region is denoted as δΩ. (b) For every point p on the contour δΩ, a patch Ψp 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, Ψq that has most similarities with Ψp. (d) The last step would be copy the pixels from Ψq to fill Ψp. 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, 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 φ_P centered at the point 'p'.We defines its priority $P(P)$ as the product of two terms:

$$P(P) = C(PD(P)$$

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

$$C(P) = \frac{\Sigma[(q \epsilon \varphi_P) \cap (I-\Omega)]C(P)}{|\Psi_P|}$$

$D(P)$The data termis a function of the strength of isophotes hitting the front ∂Ω at each iteration.

$$D(p) = \frac{\left|\nabla I_P^{\perp} \cdot \eta_P\right|}{\alpha}$$

Where, (a) n_P estimated as the unit vector orthogonal to φ_P the front ∂Ω... (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) α=Normalization factor (e.g. Gray level it is α=255)

2) Propagating Texture and Structure information:

Propagate image texture by direct sampling of the source region, the distance $\partial(\varphi_{\hat{P}}, \varphi_q)$ between two generic patches φ_P and φ_q is simply defined as the sum of squared differences.

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

3) Updating Confidence Values:

After the patch φ_P has been filled with new pixel values, the confidence $C(P)$ is updated in the area delimited by φ_P

$$C(P = C(P^{\hat{}}) \qquad \forall_{\downarrow}(P\ ) \in (\varphi_{\downarrow}P^{\hat{}} \cap T\ )$$

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 COMPARISIONS

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.This shows the drawback of divided exemplar based image inpainting on paint images but is good in real images. Result shown below.

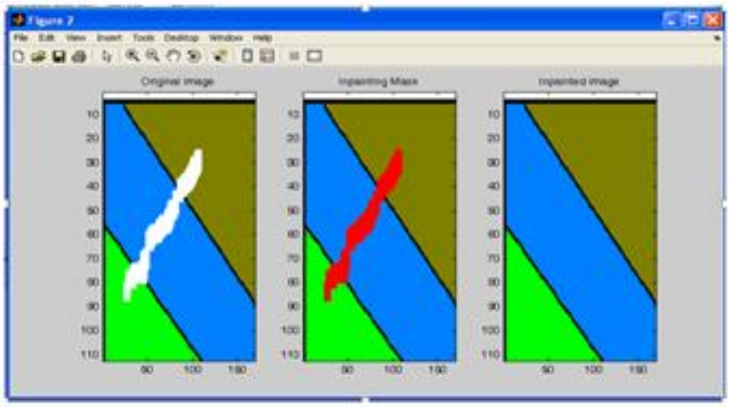


Figure 6.1 Linepaintimage.Fig 6.1 shows the result of exemplar based image inpainting where we do not divide the image and apply the algorithm then we get the good result.(a) Original Linepaint image112x169*pix*. (b) The target region to Fill (in red colour), Total no of pixels to Fill are 1095. (c) The result of region filling by our automatically Exemplar-based Image Inpaintingalgorithm, Time required to fill this target region is12.359000sec and give the good result.

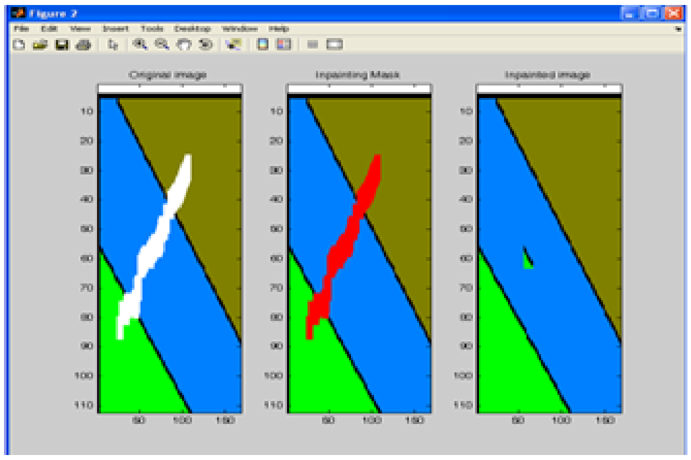
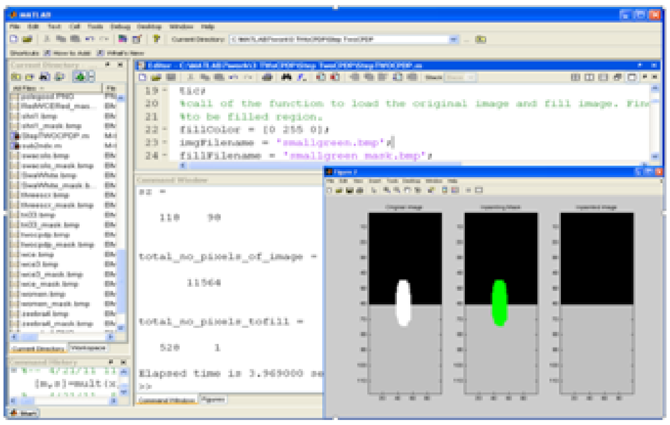


Figure 6.2 Linepaintimage.Fig 6.2 shows the result of divided exemplar based image inpainting where we divide the image and apply the algorithm but we get the result is not good..(a) Original Linepaint image112x169*pix*. (b) The target region to Fill (in red colour), Total no of pixels to Fill are 1095. (c) The result of region filling by our automatically Exemplar-based Image Inpainting algorithm, Time required to fill this target region is3.70300sec and give the bad result



```
tic;
%call of the function to load the original image and fill image. Fin
%to be filled region.
fillColor = [0 255 0];
imgFilename = 'smallgreen.bmp';
fillFilename = 'smallgreen_mask.bmp';
```

```
sz =

   118    98


total_no_pixels_of_image =

       11564


total_no_pixels_tofill =

   528     1

Elapsed time is 3.969000 se
>>
```

Figure 6.3 Smallgreen image. (a) Original Smallgreen image with white circle and image size118x98*pix*. (b) The target region to Fill (in greencolour), Total no of pixels to Fill are 528. (c) The result of region filling by our automatically Exemplar-based Image Inpaintingalgorithm, Time required to fill this target region is3.969000sec and give the good result

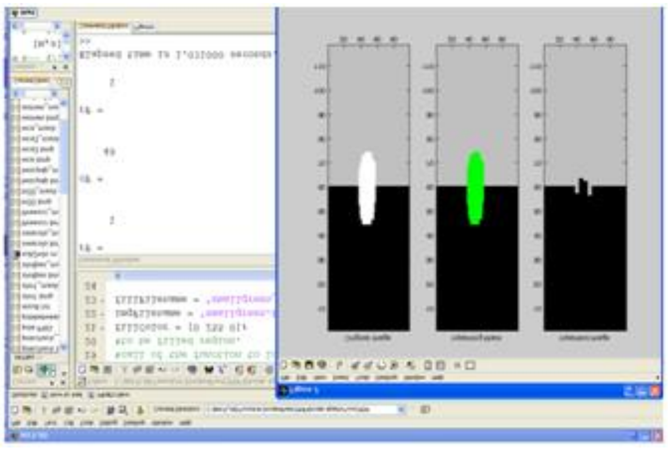
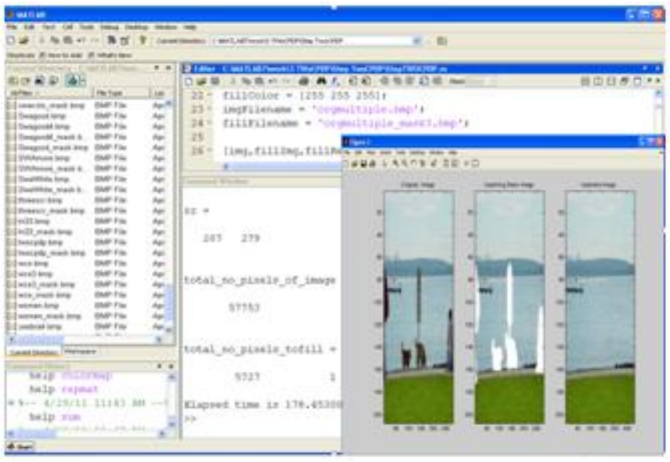


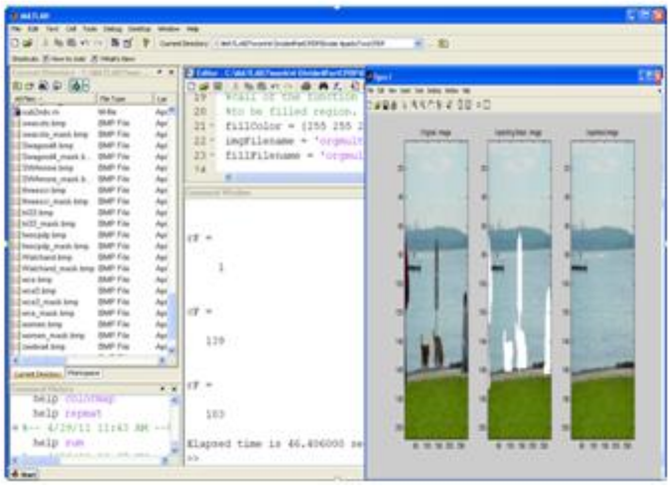
Figure 6.4 Smallgreen image. (a) Original Smallgreen image with white circle and image size118x98*pix*. (b) The target region to Fill (in greencolour), Total no of pixels to Fill are 528. (c) The result of region filling by our automatically Exemplar-based Image Inpaintingalgorithm, Time required to fill this target region is 1.031000sec and will not give the good result.



```
fillColor = [255 255 255];
imgFilename = 'orgmultiple.bmp';
fillFilename = 'orgmultiple_mask3.bmp';

[img,fillImg,fillR
```

```
sz =

   207   279


total_no_pixels_of_image

       57753


total_no_pixels_tofill =

       5727          1

Elapsed time is 178.45300
>>
```

This matlab window with output shows the result of Exemplar-based Image Inpainting algorithm apply on the original undivided image.which output is clearly shown in figure 6.5 (c) only threinpainted image.



```
%call of the function
%to be filled region.
fillColor = [255 255 2
imgFilename = 'orgmult
fillFilename = 'orgmul
```

```
cf =

    1


cf =

   139


cf =

   183

Elapsed time is 46.406000 se
>>
```

This matlab window with output shows the result of DividedExemplar-based Image Inpainting algorithm apply on the original divided image.which output is clearly shown in figure 6.5 (d) only threinpainted image using both of this algorithm the result will be nearly same but the required time to fill the target region will be different this time is good for of Divided Exemplar-based Image Inpainting.

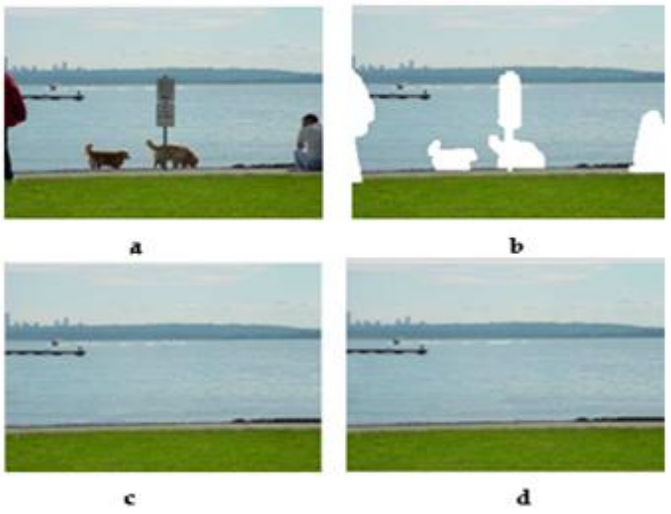
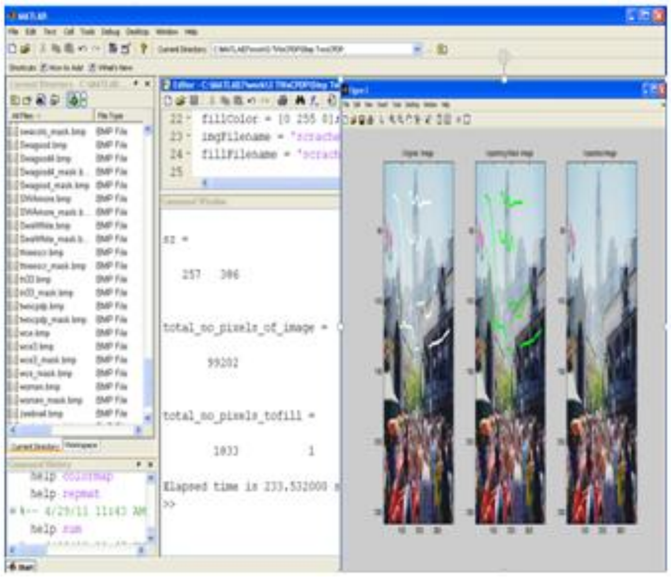


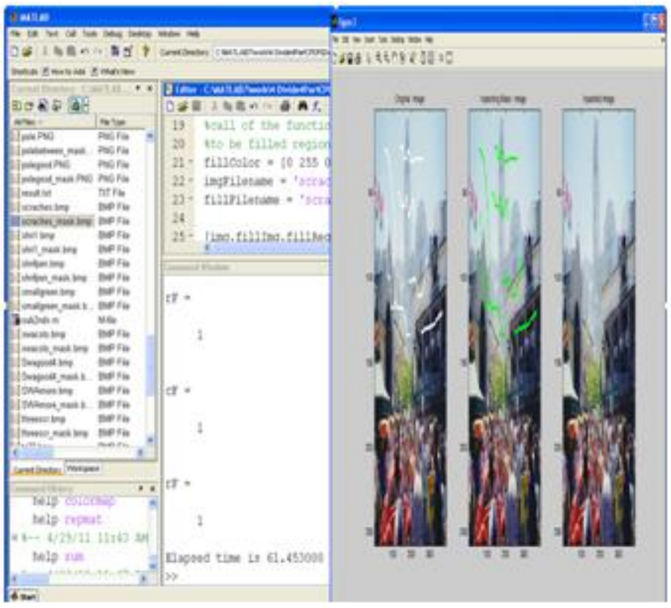
Fig 6.5: Removing multiple objects from photographs.Fig 6.3 (a) Original Sea land Image (Size 207x279) with Several objects (b) Object-Cut image(Total no. of pixels To Fill are 5727).(c) Several objects are Successfully Removed using Example Based Inpainting. Required execution time for this is 178.453000 Seconds.(d) Several objects are Successfully Removed using Divided Image in Exemplar-Based Inpainting. Required execution time for this is 46.406000 Seconds.



```
fillColor = [0 255 0];
imgFilename = 'scrach
fillFilename = 'scrach
```

```
sz =

   257   386


total_no_pixels_of_image =

       99202


total_no_pixels_tofill =

        1833           1

Elapsed time is 233.532000 s
>>
```

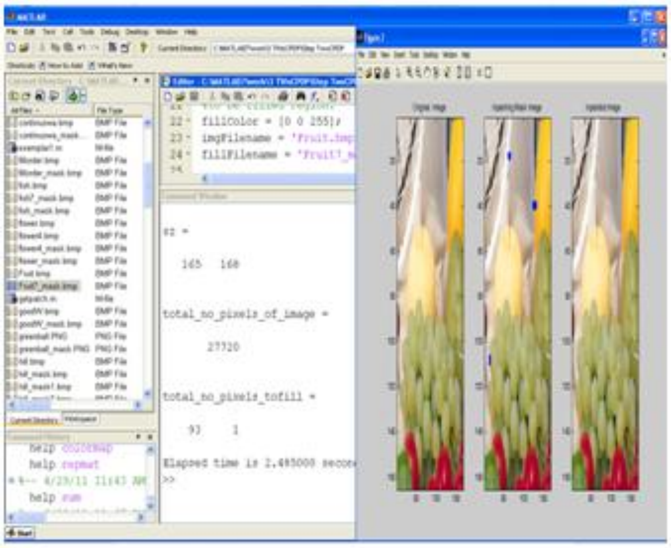
Fig6.6: Removal of Scratches(a) Original damaged Scratch Image size (257x386). (b) Mask Image (Total no. of pixels To Fill are1833.)(c)Scratches are Successfully Removed using Exemplar Based Inpainting. Required execution time for this is 233.532000 Seconds.



```
%call of the functio
%to be filled region
fillColor = [0 255 0
imgFilename = 'scrac
fillFilename = 'scra

[img,fillImg,fillReg
```

```
cF =

     1


cF =

     1


cF =

     1

Elapsed time is 61.453000
>>
```

Fig6.7: Removal of Scratches(a) Original damaged Scratch Image size (257x386). (b) Mask Image (Total no. of pixels To Fill are1833.)(c)Scratches are Successfully Removed using Divided Exemplar BasedImageInpainting. Required execution time for this is 61.453000 Seconds. Patch sizes same for both cases which are 11x11.



```
fillColor = [0 0 255];
imgFilename = 'Fruit.bmp
fillFilename = 'Fruit7_m
```

```
sz =

   165   168


total_no_pixels_of_image =

       27720


total_no_pixels_tofill =

    93     1

Elapsed time is 2.485000 secon
>>
```

Fig6.8: Fruit image a) Original Fruit Image size (165x168). (b) Mask Image using blue so in fillcoior marker it shows 255 (Total no. of pixels To Fill are 93.)(c)Target mask portions are successfully removed using Exemplar BasedImageInpainting. Required execution time for this is 2.485000 Seconds. Patch sizes same for both cases which are 11x11. Below figure 6.8.1 show the zoom out that particular target filled region indicated by red circle in original fruit image and resultant image.

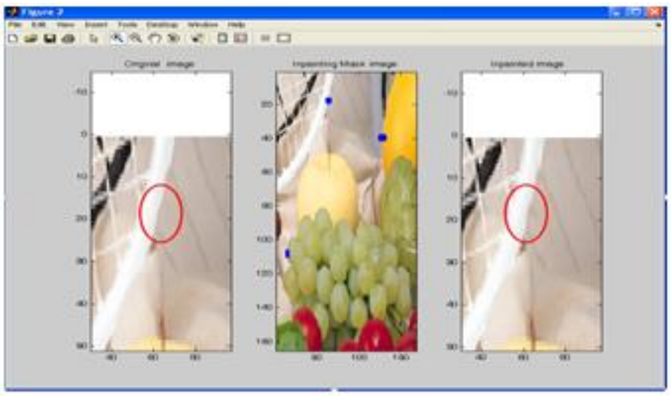


figure 6.8.1 Zoom out portion in original fruit image and after filling target region in fruit image by using surrounding information filled automatically using and it filled good but time required to fill is greater that Divided Exemplar Based Image Inpainting

srrrr

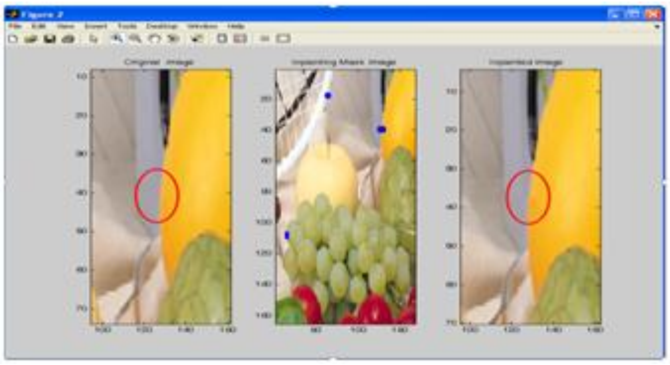
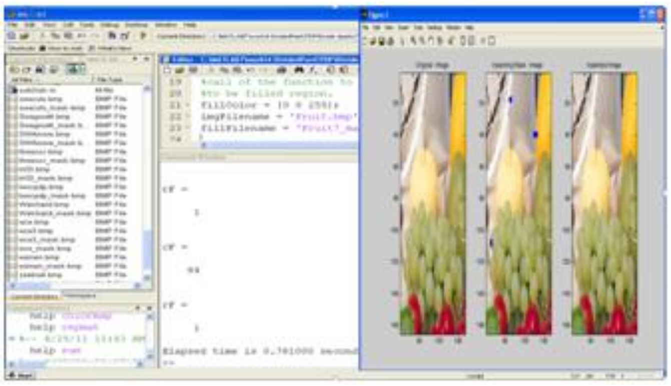
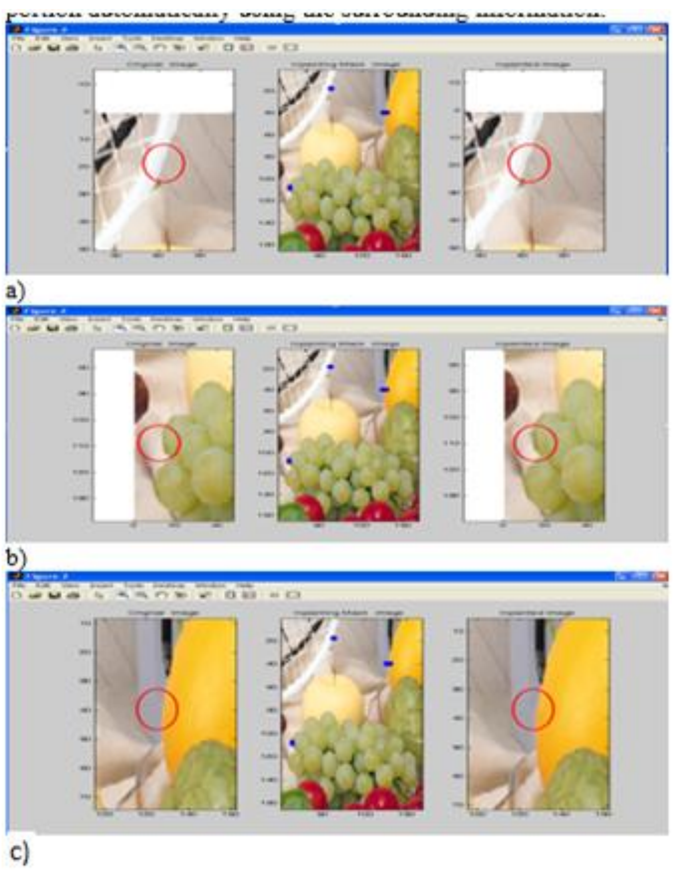


Figure 6.8.2 Zoom out another portion in original fruit image and after filling target region in fruit image by using surrounding information filled automatically using and it filled well with continuing the linear and structure and texture but time required to fill isless as compare to that Exemplar Based Image Inpainting



```
%call of the function to
%to be filled region.
fillColor = [0 0 255];
imgFilename = 'Fruit.bmp'
fillFilename = 'Fruit?_m

cF =

     1

cF =

    94

cF =

     1

Elapsed time is 0.781000 second
```

Fig6.9: Fruit image a) Original Fruit Image size (165x168). (b) Mask Image using blue so in fillcoior marker it shows 255 (Total no. of pixels To Fill are 93.)(c) Target mask portions are successfully removed using Divided Exemplar Based ImageInpainting. Required execution time for this is 0.781000 Seconds. Patch sizes same for both cases which are 11x11. Below figure 6.9.1 show the zoom out that particular target filled region indicated by red circle in original fruit image and resultant image which is filled with good texture and structure information.followinfg figure a),b) and c) shows the zoom out portion of the original image and after filled that portion automatically using the surrounding information.



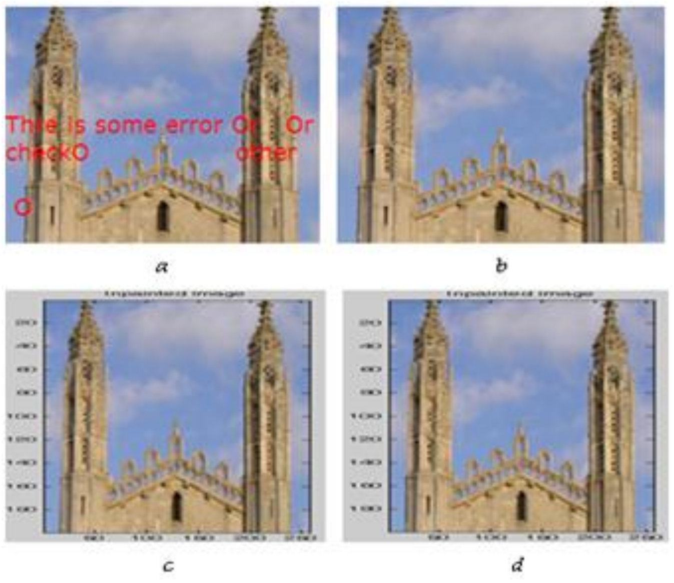


Fig 6.10: Text image(a) Original Text Image (199x257). (b) super- imposed Text Removed (Total no. of pixel to Fill are 817) (c) Result after removing Text using Exemplar Based Image InpaintingwithPatch size11x11. Required time for fill is 96.6410seconds. (d) Result after removing Text using Divided

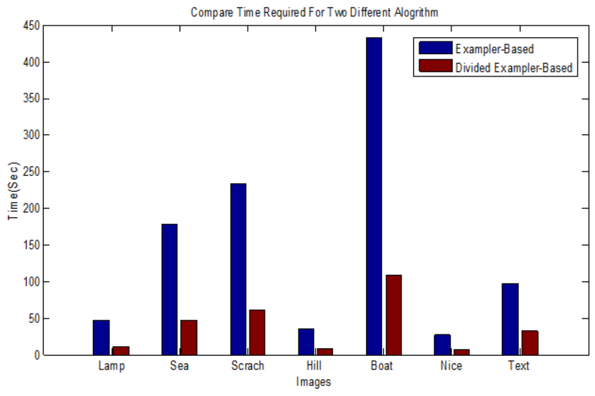
Image in ExemplarBased Image Inpainting with Patch size 11x11.Required time for fill is 32.5630seconds.

Thus by using this Divided Exemplar BasedImageInpainting OR Divided Image Exemplar BasedInpainting 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.

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

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

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.

Table 3.4: Execution time for Exemplar-Based inpainting and Divided Image in Exemplar-Based inpainting

Image	Figure	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)
Sea land	Fig 6.5	207x279	57753	9502	178.453	46.406
Scratch Image	Fig 6.6/6.7	257x386	99202	1833	233.532	61.453
Fruit image	Fig 6.8/6.9	220x176	38720	1124	27.141	7.203
Text Image	Fig 6.10	199x257	51143	817	96.641	32.563

Table 3.4 shows the also shows the time required for other images by comparing the stated two algorithm.

Ihave 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, Nodependency on image segmentation. Balanced region filling to avoid over-shooting artefacts.Patch-based filling helps achieve: (i) speed efficiency, (ii) Accuracy in the synthesis of texture (iii) accurate propagation of linear structures. Limitations of this technique: The synthesis of regions for which similar patches do not exist does not produce reasonable results. The algorithm is not very good to design curved structures. Finally, this algorithm does not handle depth ambiguities. (An expression whose meaning cannot be determined from its context) (i.e., what is in front of what in the occluded area).
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.CriminisiP.Perez and K.Toyama.“Region Filling and Object Removal by Exemplar-Based Image inpainting” IEEE Transation on image processing ,vol13, ,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.DeBonet.” 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.Olivieira, 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 forutomatedInpainting", 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] D. Garber. Computational Models for Texture Analysis and Texture Synthesis. PhD thesis, University of Southern California, 1981.

[16] **Websites:** http://www.mathworks.com/, http://www.ieeexplore.ieee.org/

[17] **Websites:** http://www.google.com/,

[18] **Reference Books:**
a) Rafael C. Gonzalez , Richard E.Woods, “Digital Image Processing ” .
b) A. K. Jain.” Fundamentals of Digital Image processing”

BIOGRAPHY
AssistantProf.Mrs.jyotiWaykule completed M.Tech. (Electronics), I have published two papers in international journal.