Region Filling and Object Removal by Exemplar-Based Image Inpainting

Manpreet Singh

Abstract—A new algorithm is proposed for removing large objects from digital images in a visually plausible way. It combines the pros of (i)"texture synthesis" as well as (ii)"inpainting" algorithms to take into account both texture and structure while filling in image gap regions. The primary approach for replicating both texture and structure is included in exemplar-based texture synthesis; however, the effectiveness of structure propagation depends heavily on the sequence in which the filling proceeds. This new algorithm propagates confidence in the synthesized pixel values similarly to information propagation in inpainting. Exemplar-based synthesis is used to compute actual colour values and computational efficiency is optimized by a block-based sampling process. Robustness of this technique in removing large occluding objects as well as thin scratches has been demonstrated by on a number of examples with varied shapes of target regions.

I. INTRODUCTION

There is an abundance of historical literature on algorithms created for removing unwanted objects from images. The purpose of object removal may be to remove unwanted objects or occlusion in order to improve the image quality. The first step in object removal is to mask out the undesirable portions of the image, leaving gaps where the object had previously been present. Texture synthesis and image inpainting are two of the most widely used techniques for removing objects from digital images and then filling in those image gaps. Both these approaches have their own positive and negative aspects. Texture synthesis might result in the loss of linear structures and image inpainting might cause blurring while dealing with complexbackgroundcomponentsorspatialstructures.

The work presented here is a novel algorithm for the removal of large objects and their replacement with realistic backgrounds in digital images. The advantages of both texture synthesis and inpainting are combined into one efficient algorithm in this approach. It emphasizes linear structures especially, similar to image inpainting. Linear structures adjacent to the target region primarily influence the fill order, essentially acting as a guide for an exemplar-based texture synthesis algorithm.
Fig.1showsanexampleofobjectremovalwherethe foreground baseball player is selected as the target region and is replaced by pixel information sampled from the remainder oftheimage.

Previous research has primarily utilized texture synthesis for filling up large image regions with repetitive 2-D textural patterns; which involves replication of texture from a small given source sample of pure texture. It effectively generates new texture by sampling and copying color values from the source if the texture is consistent, however, in photos having real-world scenes that have linear structures and composite textures interacting spatially, this technique does not perform well.

(a) (b)

Fig. 1: Removing large objects from images (a) Original image and (b) The region corresponding to the person in the foreground is extracted from the input mask and refilled by synthesized structures of the grass and ground.

The primary issue is that boundaries between different image regions are a complicated product of interactions between several textures and they form more 1-D or linear image structures. Numerous image inpainting techniques fill in image gaps by diffusing linear structures (referred to as isophotes) into the target region. Their limitation is that the blur generated bythediffusionprocessisvisiblewhenfillinglargerregions. In our new algorithm, particular attention is paid to linear structures, just like in inpainting. However, the target region's neighboring linear structures only impact the fill order of what is fundamentally an exemplar-based texture synthesis.

The final outcome is an algorithm that respects the visual limitations imposed by surrounding linear structures while simultaneously having the efficiency and qualitative performance of exemplar-based texture synthesis.

II.METHODS

An isophote-driven image-sampling method is at the core of this novel approach. Exemplar-based texture generation is sufficientforthepropagationofextendedlinearimagestructures. Isophotes are the image's linear structures, whereas composite textures are made up of the background components. To maintain the continuity of sharp edges, the algorithm must choose where to begin filling in the unknown target region

from, as filling from any random patch may result in the loss of linear structures.

Filling priority is therefore assigned to certain areas of the target zone based on two functions—Data and Confidence. Data refers to the linear structures that are considered to be in the patch, and Confidence refers to how many known pixels are in the unknown target patch within patch area. Confidence is used to prioritize patches with more source pixels.

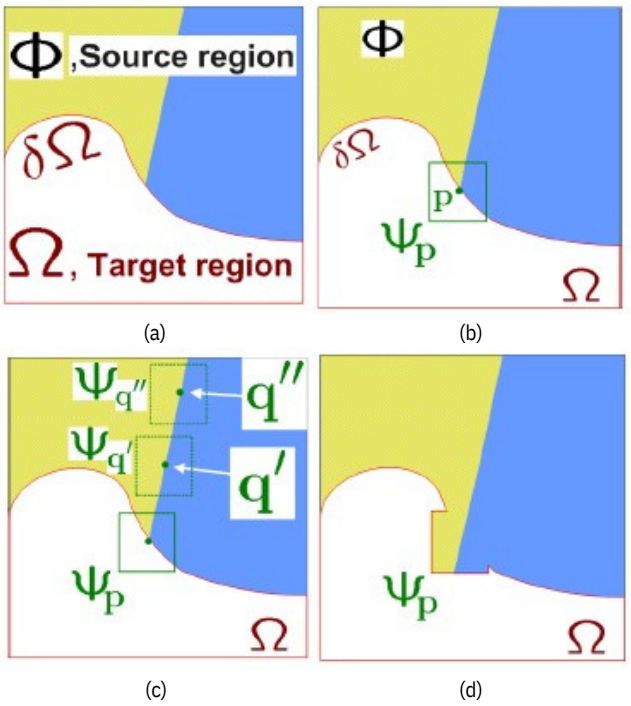


Fig. 2: One iteration in the Exemplar-Based Image Inpainting Algorithm

The region to be filled (target region) is indicated by Ω, and its contour is denoted by δΩ. As the contour grows inward while the algorithm runs, we often refer to it as the fill front. The source region Φ, which remains constant throughout the algorithm, furnishes samples for the filling process.

Here is a single iteration of the algorithm (Fig. 2):

1) Determine the contours of the target region: First, for an input image, the user chooses a target region Ω to be eliminated and filled. The source region Φ is defined as the target region subtracted from the entire image $(\Phi = I - \Omega)$, as a dilated band around the target region.

2) Specify the size of the template window: A default window size of 9x9 pixels is used, but in practice, the user is asked to set the window size to be a little larger than the largest identifiable texture element, in the source region.

3) Computing Patch Priorities: Priority values are assigned to each patch on the fill front. The priority computation favors patches that are both a) on strong edge continuations and b) surrounded by high-confidence pixels. The priority P(p) of a patch Ψp centered at point p for some $p \in \delta\Omega$ is defined as (see Fig. 3):

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

C(p) is the confidence term and D(p) is the data term, and they are defined as:

$$C(p) = \frac{\sum_{q \in \Psi_p \cap (I-\Omega)} C(q)}{|\Psi p|} \; ; \; D(p) = \frac{\nabla I^{\perp} \cdot np}{\alpha}$$

where: I = input image, Ω = target region, Ψp = current target patch, $|\Psi p|$ = area of Ψp, α=normalizationfactor, np = unit vector orthogonal to the front δΩ in the point p

C(p) is initialized as:

$$C(p) = \begin{cases} 0, & \forall p \in \Omega, \\ 1, & \forall p \in I - \Omega \end{cases}$$

For the "Data" term (D(p)), the algorithm calculates the gradient of the entire image and normalizes it with α. It measures the strength of isophotes hitting the front δΩ at each iteration. This element is crucial to the algorithm as it promotes the synthesis of linear structures first, which assures their propagation into the target region.

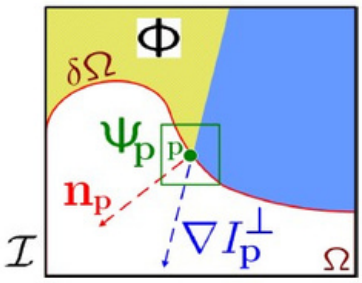


Fig. 3: Notation diagram: $\nabla I^{\perp}$ pistheisophote(directionand intensity) at point p. The entire image is denoted by I.

4) PropagatingTextureandStructureInformation:Onceall patch priorities on δΩ are computed, we find the patch $\Psi\hat{p}$ with highest priority and fill it with data from the source region Φ. Now we search in Φ for the patch most similartoour $\Psi\hat{p}$.

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

$d(\Psi\hat{p}, \Psi q)$ calculates the RGB euclidean distance between patches $\Psi\hat{p}$ and Ψq. The patch with the smallest spatial distance to the target patch is selected as the best match $\Psi\hat{q}$. The pixels in target patch are updated by copying values of corresponding pixels from $\Psi\hat{q}$.

5) Updating confidence values: The Confidence (C(p)) is now updated for $\Psi\hat{p}$ as

$$C(p) = C(\hat{p}) \quad \forall p \in \Psi\hat{p} \cap \Omega$$

This measures the relative confidence of patches on the fill front. We locate the patch that surrounds each pixel,

keeping the pixel in the center. Then, the confidence level for each pixel is determined by dividing the total number of known pixels by the patch size.

Implementation details: In this algorithm's implementation the contour $\delta\Omega$ or boundary of the target region is represented as an extensive list of image point locations (coordinates). These points are extracted by using a laplace filter on the input mask.

Then the normal direction np for a point p $\in \delta\Omega$ is computed:

1) Using a two-dimensional Gaussian Kernel, the 'control' point positions of $\delta\Omega$ are filtered

2) Unit vector np is computed as the unit vector perpendicular $\perp$ to the line connecting the immediately preceding and following points in the list.

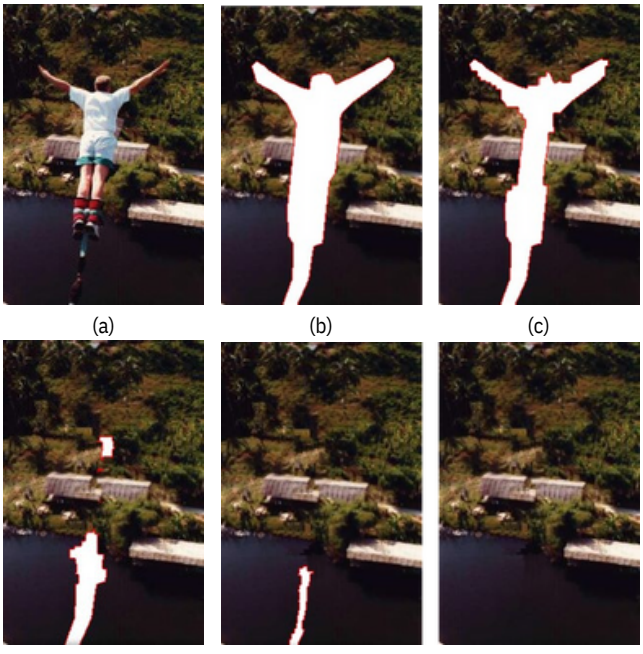
Gradient ∇I_p is the maximum value of the image gradient Π. Finally, depending upon the value assigned to the alpha Ψp

component of a pixel, it is categorized as belonging to the target region Ω or source region Φ or the remaining portions of the image.

III. RESULTS

Here we display the outputs of our algorithm applied to a few images and their corresponding masks. In all of the cases, it is evident that this algorithm extends the contour inwards more naturally between different textures while reconstructing the target region.

Fig. 4 shows that while the algorithm runs, thinner areas (such as the arms) in the target region tend to disappear faster as compared to the thicker areas, and isophotes hitting the boundaries of the target region spread inward. In the final result, we can see that there is no visible blur and high- frequency textural information is present in the reconstructed region.

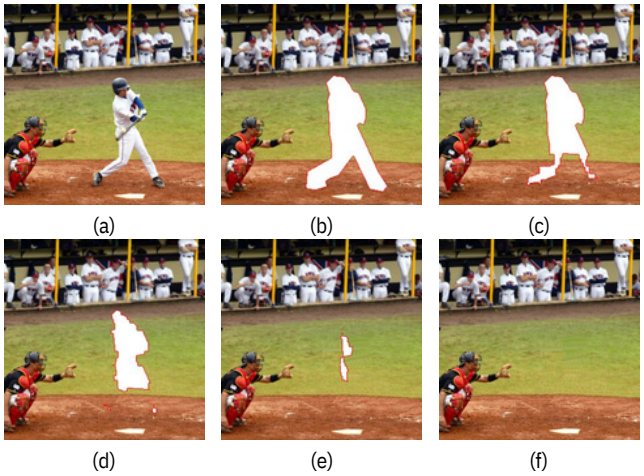


(a) (b) (c)

(d) (e) (f)

Fig. 4: Removing large objects from images (a) Original image (b) The target region with red boundary (c), (c), and (d) Stages of filling up target region (f) Final result - bungee jumper removed completely and occluded background reconstructed

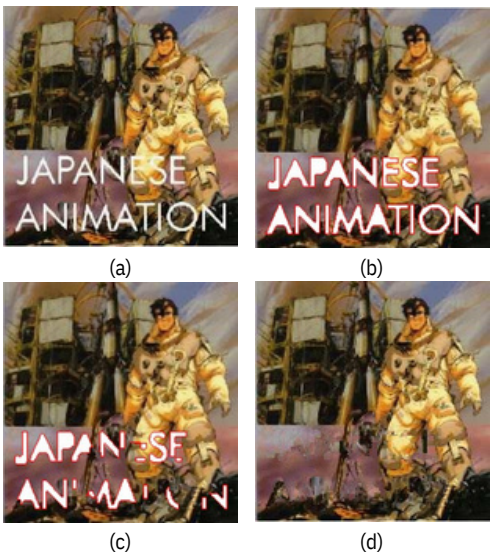
Fig. 5 shows the removal of the baseball player from the foreground. Here our algorithm perfectly fills in the target region with appropriate textural information from the back- ground and accurately preserves the linear structures. This is due to an abundance of sample data available in the original image around the target region for refilling.



(a) (b) (c)

(d) (e) (f)

Fig. 5: Removing person from image (a) Original image (b) The target region with red boundary (c), (c), and (d) Stages of filling up target region (f) Final result - baseball player removed completely and occluded background reconstructed



(a) (b)

(c) (d)

Fig. 6: Example of image restoration (a) Original image (b) The target region with red boundary (c) Intermediate stage of restoration (c) Final result-after text removal

In Fig. 6, our goal is to remove the text and recover the underlying image. This example highlights that our algorithm achieves identical results as with the image inpainting algorithm. Thus our algorithm performs both texture synthesis as well as inpainting efficiently in image restoration applications.

IV. DISCUSSION

A novel algorithm has been presented in this study that efficiently removes large objects as well as minor scratches from digital photographs and reconstructs the background. This algorithm uses the texture synthesis approach and includes the inpainting techniques to improve the background regeneration.

The advantages of this technique can be listed as:

1) Preservation of image sharpness
2) Over-shooting artefacts are avoided due to the presence of the Confidence term in the Priority function. The priority function estimates the desirable filling order of pixels in the target region. As filling progresses, pixels in the target region's outer layers will have higher confidence values and thus be filled sooner; pixels in the target region's center will have lower confidence and thus be filled later.
3) Algorithm does not depend image segmentation
4) 2-D texture synthesis performed efficiently
5) Accuracy in the propagation of linear structures into target region

Also, there are few limitations of this algorithm as discussed below:

1) Unabletoproduceaccurateorreasonableresultsifthere is a lack of sufficient examples of the target region in the background.
2) Algorithmisnotdesignedtodealwithcurvedstructures
3) Algorithm does not handle image depth ambiguity. For example, a similar patch may be found that was physically at a different depth than the target patch.

Fig. 7 shows the cases where our algorithm regenerates the target region background with some artefacts due to a lack of availability of similar patches near the red boundary. Also, there is some visible blurring at patch sizes 9 and 13. Our algorithm is further limited to dealing with linear structures only, so it is unable to propagate curved structures for filling the target region.
Similarly, we see artefacts in the final result of the bungee-jumper image (Fig. 4f). The roof-like structure in the center of the image contains some artefacts due to the lack of sufficient identical sample data in the original image. It reconstructs some portion of the linear structures correctly and maintains the same orientation of the isophotes as they propagate inwards, but more sample data is required for higher accuracy.

Fig. 8 shows an example of our algorithm working on curved structures. As seen in the reconstruction, our algorithm

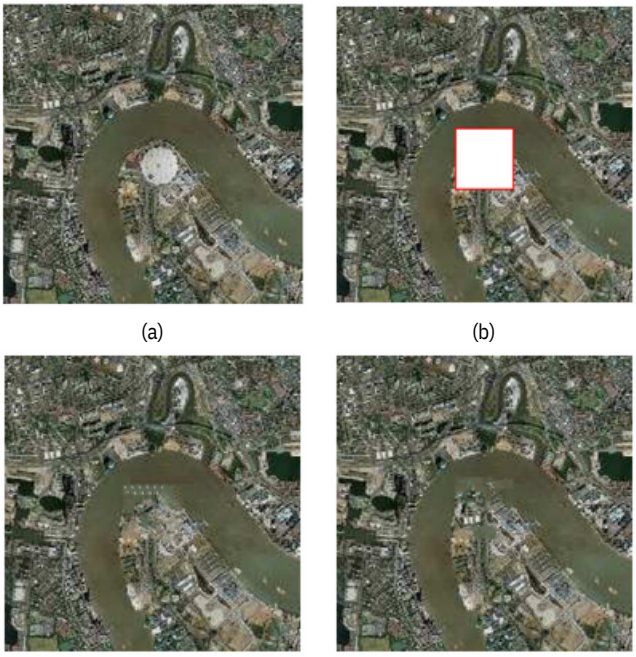


Fig. 7: (a) Original image (satellite view of London) (b) The target region with red boundary (c) Filling with patch size=9 causing artefacts, (f) Filling with patch size=13 causing blur.

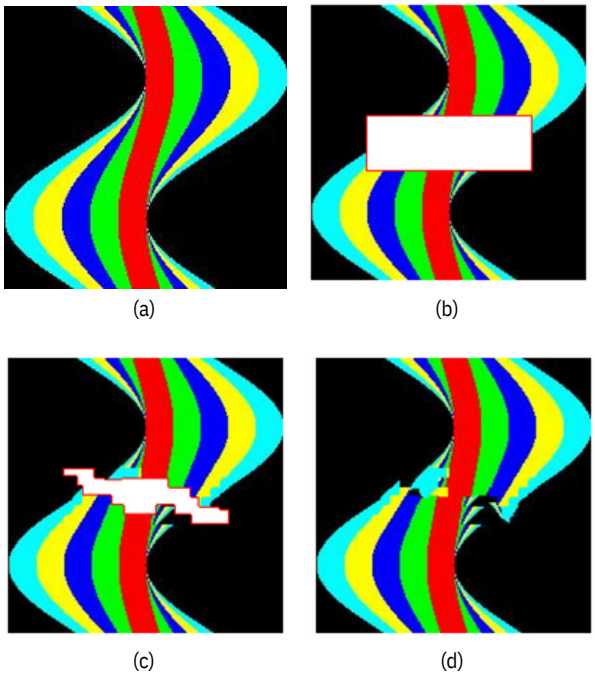


Fig. 8: Example showing how curved structures cannot be reconstructed with our algorithm (a) Original image (b) The target region with red boundary (c) Intermediate stage showing the filling process (d) Final result-after target region filling

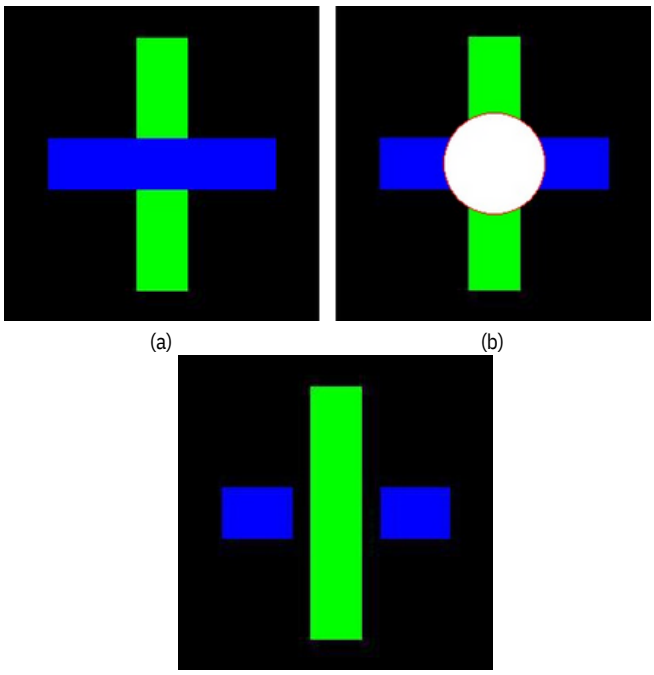


Fig. 9: Example showing that algorithm cannot handle depth ambiguity (a) Original image (b) The target region with red boundary (c) Final result-after target region filling. The algorithm cannot say whether green overlaps blue or blue overlaps green.

propagates the linear structures into the target region marked by the red boundary. The curved lines cannot effectively propagate inwards and create the original image.

In Fig. 9, our algorithm cannot reproduce the original image even when sufficient information is present. It fills up the target region by assuming similar patches are found at a physically different depth than the target patch.

Currently, research is being conducted to efficiently handle curved structure propagation and improve texture generation for robust object removal in images as well as videos. These are difficult tasks because they require high computational efficiency, more ground truth data, and datasets for performance evaluation.

REFERENCES

[1] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," IEEE Transactions on Image Processing, vol. 13, no. 9, pp. 1200-1212, 2004. doi:10.1109/TIP.2004.833105.

[2] H. Park, A Comparative Analysis of Exemplar-Based Image Inpainting (EBII) and Simultaneous Cartoon and Texture Image Inpainting Using Sparse Representations, http://www-personal.umich.edu/ hyunjinp/proj/sample/sample full report.pdf

[3] Ye Hong, Object Removal Using Exemplar-Based Inpainting, https://pages.cs.wisc.edu/ yhong/yhong report.pdf