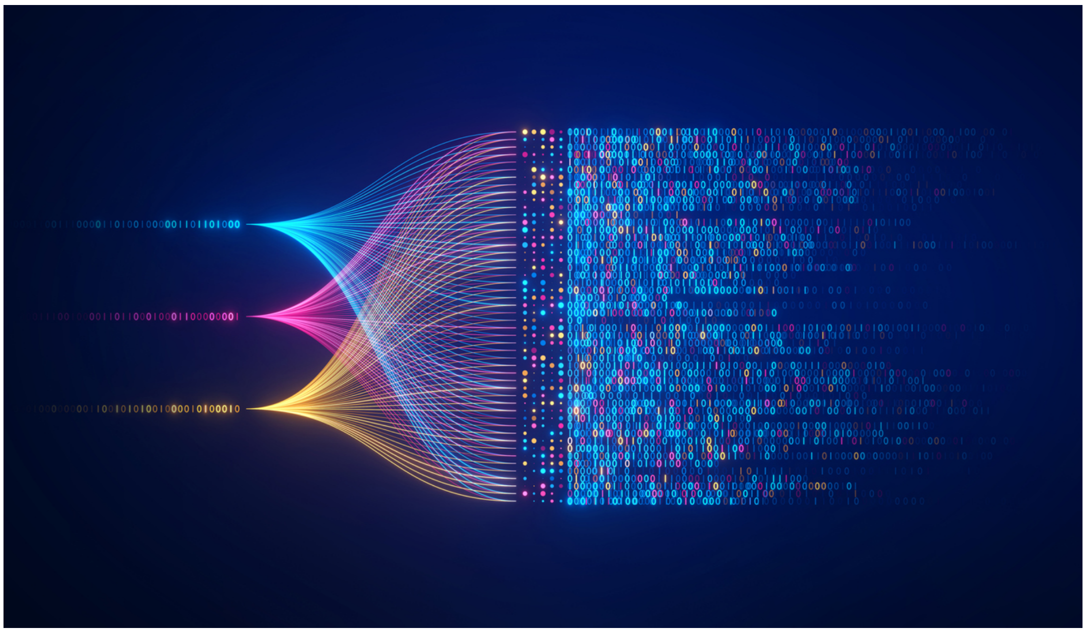
Region Filling and Object Removal by Exemplar-Based Image Inpainting



INTERNSHIP REPORT

Submitted by :- Swagata Naskar
Electronics and Electrical Communication Engineering
Indian Institute of Technology Kharagpur

Project Guide :- Dr. Tanmay Dutta (IIT Guwahati)

Date Of Submission :- 19th Dec, 2022

Contents

1 Introduction

Problem Statement: We are asked to implement ***Criminisi et al.* (2004)** paper. The paper presented a novel algorithm for large object removal and filling the hole left behind in a visually plausible way.

2 Literature Survey

In the past, there were two types of algorithms which addressed these issues in two different types of situations. i)**Texture synthesis** algorithms for generating large image regions using samples. ii)**In-painting** techniques for small image gaps. The former type is helpful in generating two dimensional textures. The latter helps with linear structures such as lines and contours. The algorithm presented in this paper is a single efficient approach for simultaneous propagation of both texture and structure. The paper states that the exemplar based texture synthesis is sufficient for propagation of both texture and structure. However, the order in which the filling proceeds highly influences the structure propagation. In this algorithm, **confidence**(a term which will be explained later) in the synthesized pixel values is propagated in a similar manner to the propagation of information in in-painting. However, the filling of actual color values of the pixels will be done using exemplar based synthesis.

In the in-painting techniques, inspired by the partial differential equations of heat flow, isophotes/linear structures are propagated into the target region via diffusion in order to fill the holes. Those techniques are not of much help when filling large regions as diffusion introduces noticeable blur. Therefore, in this algorithm, the foreground(which is to be manually selected as the target region) is automatically replaced by the data sampled from the remainder of the image. This is inspired by the exemplar based synthesis. The takeaway from the in-painting is that the linear structures adjacent to the target region influence the filling order. The regions in which there is more reliable information and which are more crucial for propagating linear structures will be filled first. For that, in the algorithm in addition to the color values of the pixels, confidence values are given. Filling operation happens patch wise and each patch will be assigned a priority value which depends upon a **"confidence term"** and a **"data term"**. Thus, the paper combines the strengths of both of these approaches to present a novel and efficient algorithm. Using patch based filling instead of the pixel based filling improves both speed of execution and the accuracy of the propagated structures. The performance of the algorithm is robust with respect to the shape of the manually selected target region.

3 Algorithm

For a given input image, user first selects the target region Ω to be removed and filled to suit the background. This background (Φ) or ($\mathbf{I}$-Ω) would be called the source region. The boundary of the target region is called contour/fill-front and is represented by $\delta\Omega$. As this is a patch based filling process, patch size must be specified. Generally, it is taken as 9×9. However it is a better idea to take the patch size to be slightly bigger than the largest distinguishable texture element(texel). As soon as the target region is selected, a mask(binary image) is made in which $\forall$ pixels p $\in \Omega$, 1 is assigned, and the rest of the image, 0. And then a new image is created where target region Ω is white-washed. We fill the colour values in this image and display the output.

During the course of algorithm, every pixel in the image will be assigned a **confidence** value which shows how confident we are in the colour value of pixel. We also calculate and assign priority value for patches centered on the pixels along the contour. The patch with the highest priority value will be filled first. The priority computation is biased toward those patches which: (i) are on the continuation of strong edges and (ii) are surrounded by high-confidence pixels. Now the following three steps will be looped until the required output is obtained.

i)Computing patch priorities: For a patch Ψ_p centered on a pixel p $\in \delta\Omega$, the **priority $\mathbf{P(p) = C(p)D(p)}$**. Here C(p),D(p) are called **confidence term, data term** respectively. Confidence of a pixel c(q) is set to be 0 $\forall$ q $\in \Omega$ and 1 otherwise in the beginning. The confidence term C(p), data term D(p) for a patch centered at pixel p are given by:

$$\mathbf{C(p)} = \frac{\sum_{q\in\Psi_p\cap(I-\Omega)} c(q)}{|\Psi_p|}$$

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

Here $|\Psi_p|$ is the patch size. Here $\nabla I_p^{\perp}$ stands for the image gradient vector which gives the isophote direction and intensity at that point. We considered the point in the patch at which there is the highest intensity to represent the patch. n_p is the unit normal vector at the pixel p. α is the normalisation factor.

ii)**Filling the patches:**
Once all the priorities of the patches on the boundary are computed, we proceed to start filling the patch with the highest priority. As we discussed earlier, we are not going to use diffusion for this purpose. We copy the information directly from the "ideal sample" in the source region. We pick the sample which has the minimum euclidean distance with respect to the patch that needs to be filled.

It has to be noted that the distance is to be calculated only for the pixels that are already filled in the patch that needs to be filled. Once we found such an exemplar patch in the source region, for each pixel that needs to be filled, we copy the value from it's corresponding position inside the exemplar patch.

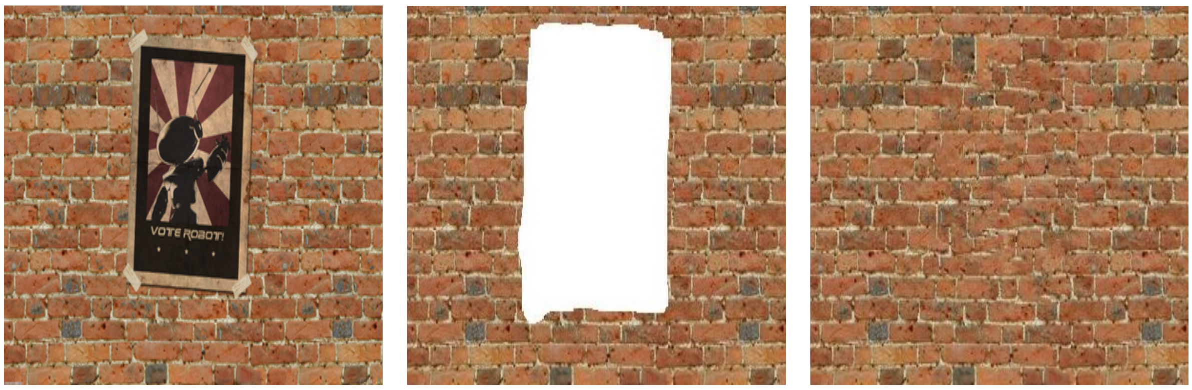
iii)**Updating the confidence values:**
c(p) = C(p') $\forall$ p $\in \Psi_{p'} \cap \Omega$
Here c(p) stands for confidence value of an individual pixel whereas C(p') stands for the confidence term evaluated for the patch centered at p' which lies between 0 and 1. It is to be noted that the confidence values are only being updated for newly filled pixels. Once filled, the confidence value of a pixel freezes. Thus, as the filling proceeds, confidence values decay denoting that we are less sure of the values that are filled at the center of target region Ω.After this the contour will be updated.

Pseudo-code is provided below:

```
• Extract the contour drawn by the user
• Repeat until done:
    1. (a) Identify the contour $\delta\Omega^t$
       (b) Compute the priorities P(p) $\forall p \in \delta\Omega^t$
    2. (a) Find patch $\Psi_{p'}$ with maximum priority.
       (b) Find the exemplar patch $\Psi_{q'}$ from source region with minimum
           distance to $\Psi_{p'}$.
       (c) Copy the pixel values from $\Psi_{q'}$ to $\Psi_{p'}$ $\forall p \in \Psi_{p'} \cap \Omega$
    3. Update c(p) $\forall p \in \Psi_{p'} \cap \Omega$.
```

4 Results

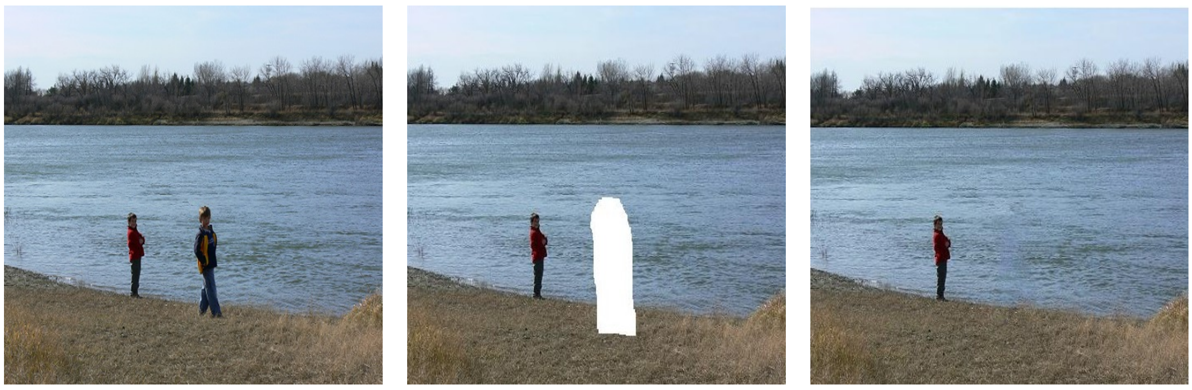
4.1 Bricks



(a) (b) (c)

Figure 1: (a)actual image (b)mask (c)final output and the alignment of the bricks has been restored to a large extent.

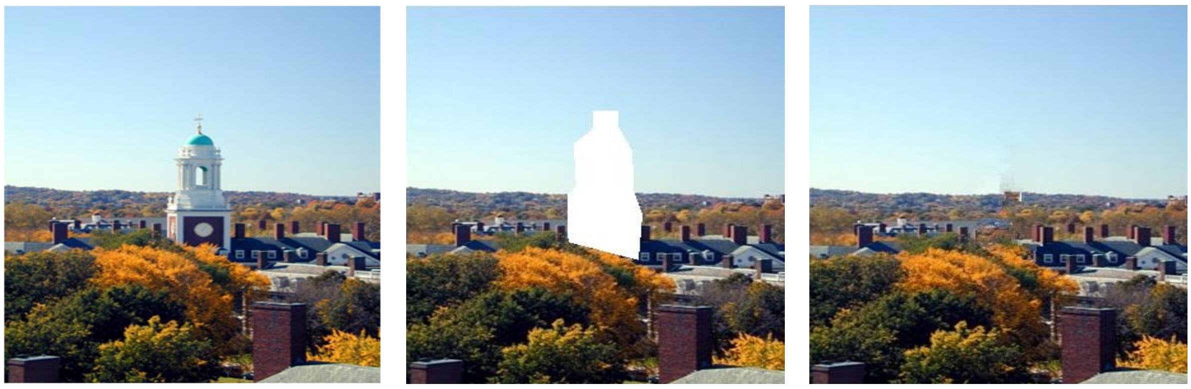
4.2 River



(a) (b) (c)

Figure 2: (a)actual image (b)mask (c)final output and the land and water boundary has been successfully reconstructed.

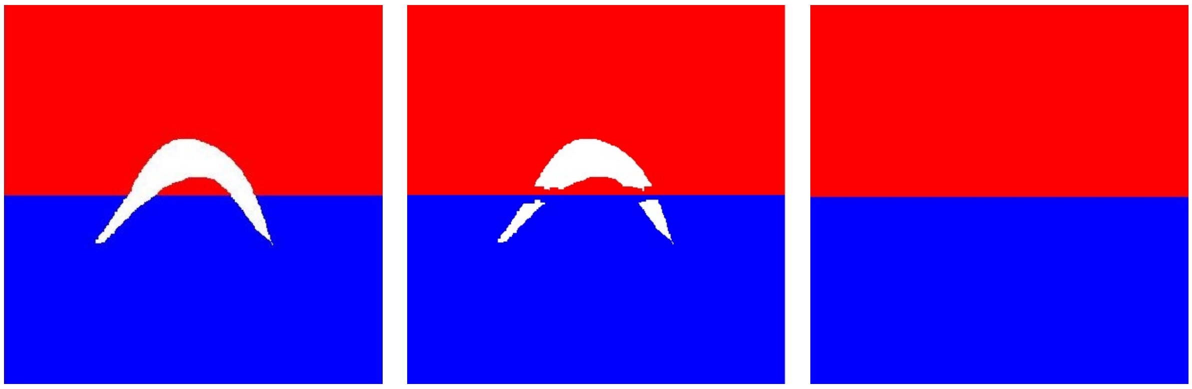
4.3 Tower



(a) (b) (c)

Figure 3: (a)actual image (b)mask (c)final output and the sky and tree regions have been nicely propagated though a small noise pixel has been observed.

4.4 Half Moon



(a) (b) (c)

Figure 4: (a)mask (b)intermediate image (c)final output and filling the target region by an edge-driven filling order achieves the desired output.

4.5 Colors

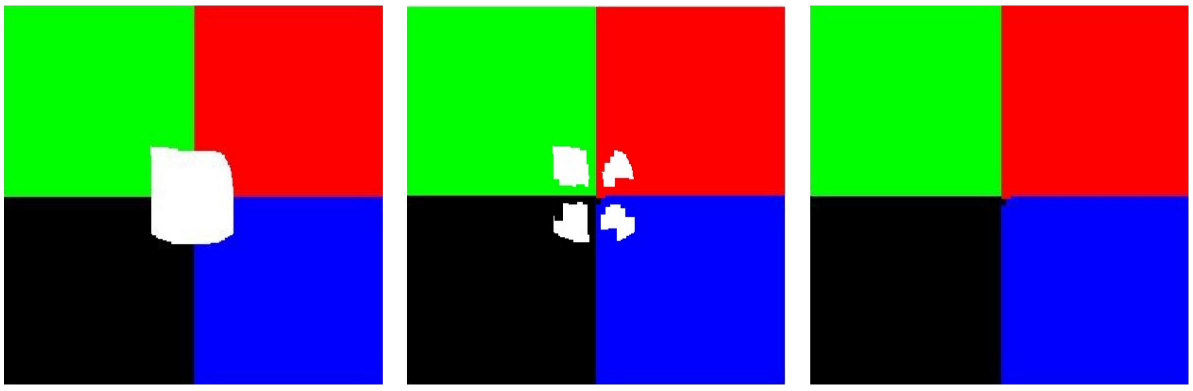
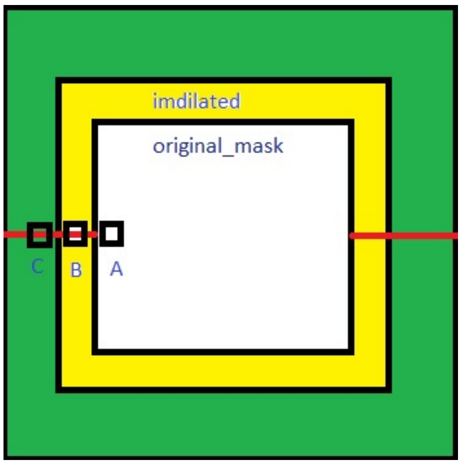

(a) (b) (c)

Figure 5: (a)actual image (b)mask (c)final output and observed the propagation rates of different boundary lines.

5 Discussion



1. Green color and yellow color together are the source region and the line shows the direction of the isophote. For calculating the isophote value and direction of a pixel 'A' on the contour, we will consider a patch of size n*n centered at the pixel 'A', and the maximum isophote value in that patch is assigned as the isophote value of 'A'. We will not consider the pixels in the yellowed shaded region to avoid the gradient occurring due to color change by our original mask. The imdilated mask size is less than the patch size at pixel A.

There could be a steep gradient near the boundary of the target region since we whitewash the target region. So we performed imdilate operation on the mask and when calculating the data term of patches, we didn't consider the gradient values at the pixels which are present in the vicinity of the boundary. This improved our results

2. In the case of Figure 5, we can clearly see that the patches at sharp edges are being filled first. This is because, as the paper already suggested, the confidence term is biased towards sharp edges.

3. In figure 5(c), Since the data term is biased towards the patches where the isophote values are higher and places where isophotes flow into, the patches present near the boundaries with high gradient were filled first. But as we reach further into the target region, confidence decays and the patches near other boundaries get filled.

4. If we happen to find multiple patches that are equidistant from the patch which is to be filled, we consider the patch with the least gradient value at its center to be the exemplar patch. This can happen especially when images are not natural.Ex: Flags,designs etc. This helped us statistically with propagating linear structures.

5. First we used uint8 format for the image when we were trying to find the exemplar patch from the source region. But, negative numbers were typecasted as zeros on the LHS. So, we were not able to calculate the distances accurately. We finally debugged and changed the format to double and got the results right.

6. The paper took into consideration the maximum isophote value within the patch into consideration when calculating the data term(normal.*isophote). Our results were getting noisy when we tried implementing that, but directly computing the data term for all pixels and then taking into consideration the maximum value helped us.

We have attached both the code files considering patch size 9*9
Code1 ——— considering max isophote
Code2 ——— considering max data values