Implementation of the "Region Filling and Object Removal by Exempler-based Image Inpainting" paper by Criminisi, Perez, Toyama

Shubham Sahoo
17EC35023

Soham Saha
17EC35026

Pravartya Dewangan
17EC35041

Tias Mondal
17EC35043

October 24, 2020

Introduction

This paper basically implements an Algorithm that demonstrates an efficient image inpainting method. The term "Image Inpainting" can be defined as the process in which the damaged, deteriorated and missing parts of an image is filled to reconstruct the complete image. Here, in this Algorithm, firstly a target object from an image is identified and then the image inpainting technique fills the target area by the data sampled from the remaining parts of the image.

This Algorithm basically performs an efficient combination of the two techniques : texture synthesis algorithms for the generation of large image regions and the image-inpainting algorithm to fill the smaller image gaps. The exemplar-based texture synthesis is very crucial for the replication of texture and structure. Throughout the whole process, a block based sampling process is used in which a patch size is decided first. After this, some essential values like the confidence terms and data terms are calculated that play an important role in the best patch determination and texture structure propagation.

Approach

There are lots of algorithms already implemented for region filling problems.One of the algorithm used a technique for filling image regions based on a texture-segmentation step and a tensor-voting algorithm for the smooth linking of structures across holes.But the drawback is :
• requires expensive segmentation step
• requires a hard decision about what constitutes a boundary between two textures

This approach avoids both issues through the use of a continuous parameter based on local gradient strength only.
The two key observations which form the basis of this algorithm:

Sufficiency of Exemplar-based synthesis : The core of this algorithm is based on isophote-driven image sampling process. It is well understood that a separate synthesis mechanism is not required for handling isophotes. The region to be filled is called the target region. The contour of the target region evolves inward as the algorithm progresses. The source region remains fixed throughout the algorithm and provides samples used in the filling process.To improve the execution speed and accuracy of the propagated structures, patch-based filling is done instead of pixel-based.

Crucial importance of Filling Order :
Different strategies of filling order concluded to
• a better filling algorithm would be one that gives higher priority of synthesis to those regions of the target area which lie on the continuation of image structures
• is finding a good balance between the propagation of structured regions and that of textured regions

Region-Filling Algorithm

Computing patch priorities

Given a patch Ψ_p centred at the point p for some $p \in \Omega$ we define its priority P(p) as the product of two terms:

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

$$C(p) = \frac{\sum_{q \epsilon \psi_p \cap (I-\Omega)} C(q)}{|\Psi|}\ , \qquad\qquad D(p) = \frac{|\nabla I_p . n_p|}{\alpha}$$

C(p) is the confidence term which is a measure of the amount of reliable information surrounding the pixel p and the those patches will be filled first. As filling proceeds, pixels in the outer layers of the target region will tend to be characterized by greater confidence values, and therefore be filled earlier; pixels in the centre of the target region will have lesser confidence values.

D(p) is the data term which is a function of the strength of isophotes hitting the front $\delta\Omega$ .This factor is of fundamental importance in our algorithm because it encourages linear structures to be synthesized first, and, therefore propagated securely into the target region.

Propagating texture and structure information

We fill the highest priority patch with data extracted from the source region Φ. We search in the source region for that patch which is most similar to the current highest priority patch selected.

$$\Psi_{\tilde{q}} = arg\ min\ d(\Psi_{\tilde{p}}\ ,\ \Psi_q)$$

where d is the distance between two generic patches and the definition of measuring **distance,d** is mentioned in the next section.

Updating confidence values

After the patch $\Psi_{\tilde{p}}$ has been filled with new pixel values, the confidence C(p) is updated in the area delimited by $\psi_{\tilde{p}}$ as follows:

$$C(p) = C(\tilde{p}) \qquad \forall p \in \Psi_{\tilde{p}} \cap \Omega$$

This update allows us to measure the relative confidence of patches on the fill front, without image-specific parameters.

Implementation Steps:

• Extract the manually selected initial front $\delta\Omega^0$.
• Repeat until done:
1a. Identify the fill front $\delta\Omega^t$. It $\Omega^t = \phi$, exit.
1b. Compute priorities P(p) $\forall p \in \delta\Omega^t$.
2a. Find the patch with the maximum priority, $\Psi_{\tilde{p}}$
2b. Find the exemplar $\Psi_{\tilde{q}} \in \Phi$ that minimizes $d(\Psi_{\tilde{p}}\ ,\ \Psi_{\tilde{q}})$
2c. Copy image data from $\Psi_{\tilde{q}}$ to $\Psi_{\tilde{p}}$ $\forall p \in \Psi_{\tilde{p}} \cap \Omega$
3. Update C(p) $\forall p \in \Psi_{\tilde{p}} \cap \Omega$

Our New Approaches for Distance Calculations

Sum of Squared Differences - L2 distance

This is the distance that has been used in the paper.

$$d = \sum\sum(q-p)^2 \qquad\qquad \forall p \in \Psi_{\tilde{p}}, \forall (q \in \Psi_q) \in \Phi$$

It computes the error between two patches by squaring the differences between two corresponding pixels and summing them all. It is best suited for grayscaled images, as the pixels are present in 1D space. The first summation operator is for the three colour spaces and the second is for the patch dimension.

Sum of Absolute Differences - L1 distance

We have experimented this distance in our algorithm.

$$d = \sum\sum|(q-p)| \qquad\qquad \forall p \in \Psi_{\tilde{p}}, \forall (q \in \Psi_q) \in \Phi$$

It computes the error between two patches by taking the absolute value of the differences between two corresponding pixels and summing them all. This also gave good results but only in grayscaled images, however SSD (method -1) is preferred more over this. The first summation operator is for the three colour spaces and the second is for the patch dimension.

Sum and Product of Absolute Differences - Used in our Algorithm

The drawbacks of previous two algorithms are :
1) They just minimise the distance in a single colour space and do not take into account the amount of distance dissimilarities between different colour spaces.
Example :- P1 (170,200,150), P2 (80,245,40), P3 (255,255,255)
Here P1 is the Patch to be replaced by either P2 or P3.
The L2 distance between P1 and P2 = 22225
The L2 distance between P1 and P3 = 21275
The L1 distance between P1 and P2 = 245
The L1 distance between P1 and P3 = 245
Our proposed method's distance between P1 and P2 = 445745
Our proposed method's distance between P1 and P3 = 491120

If we visualize the pixels in RGB colour space, then P1 is light green, P2 is green and P3 is white. One would expect that P1 is mapped to P2. But as we can see in the above calculations that P1 is more close to P3 in L2 distance and equally distant in L1 distance. Hence we conclude that L2 or L1 distance alone is not sufficient for distance calculations.

So we finally choose this distance for our algorithm.

$$d = \sum\sum |(q-p)| + \prod\sum |(q-p)| \qquad \forall p \in \Psi_{\tilde{p}}, \forall (q \in \Psi_q) \in \Phi$$

It computes the error between two patches by taking the absolute value of the differences between two corresponding pixels and summing them. It also takes the product of the distance calculated in the L1 space for individual colour spaces. This assumption is based on the fact that more prominent colour space should have more weightage in distance calculations. This algorithm gave us the best results among the three of them. The first summation/product operator is for the three colour spaces and the second is for the patch dimension.

Results

Following are the results using our algorithm.

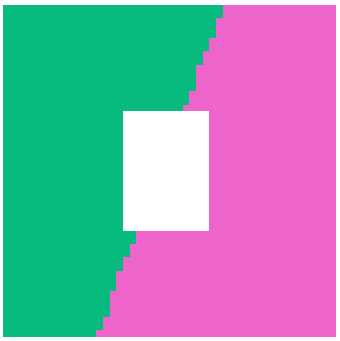

Figure 1: Input Image

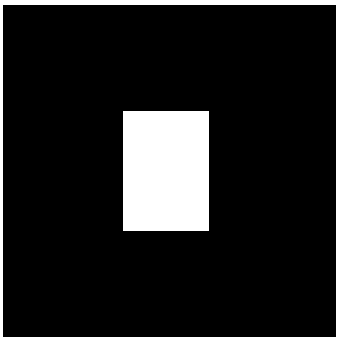

Figure 2: Mask Image

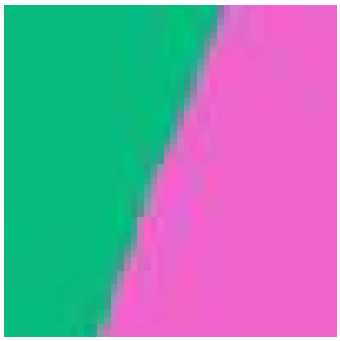

Figure 3: Output Image

Figure 4: Input Image

Figure 5: Mask Image

Figure 6: Output Image

Figure 7: Input Image

Figure 8: Mask Image

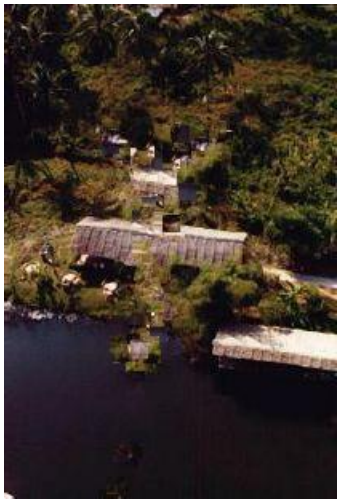

Figure 9: Output Image

Figure 10: Input Image

Figure 11: Mask Image

Figure 12: Output Image

Figure 13: Input Image

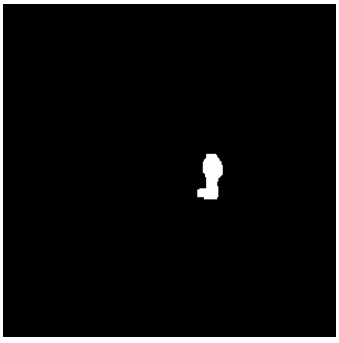

Figure 14: Mask Image

Figure 15: Output Image

Figure 16: Input Image

Figure 17: Mask Image

Figure 18: Output Image

Figure 19: Input Image

Figure 20: Mask Image

Figure 21: Output Image

Figure 22: Input Image

Figure 23: Mask Image

Figure 24: Output Image

Discussion

1) The size of the patch that we choose depends on the image that is operated on. If the image is more detailed or we need a finer output image with less distortions in colour then the patch size should be selected a bit lesser than the normal. The mask image also influence the patch size, as smaller area masks require smaller patch sizes.

2) Choosing the right distance calculation method is of great importance. In all of the images our method outperforms the traditional SSD distance. We can also use a weighted kernel of different distances in Lp norm, but again this would increase the time complexity of the algorithm.

3) There was a little discomfort while associating the isophotes with the respective pixels. So a better and robust approach is to diffuse the isophote information to the 8-neighbour pixels in a weighted manner. This would slightly relief the error caused by the quantization of the image.