Implementation of Inpainting Using Exemplar-Based Texture Synthesis

1st Ashvin Pidaparti *University of Minnesota*
Minneapolis, United States of America
pidap008@umn.edu

***Abstract*—This is an implementation of the algorithm put forward by Criminisi et al. Their algorithm posits a new method of inpainting images while preserving linear structures via exemplar matching and a new mechanism for determining the fill order of pixels. While their results could not be reproduced, we evaluate their methods and the limitations of their methods as a novel algorithm for inpainting. While texture synthesis techniques attempt to fill regions by replicating textures and diffusion techniques propagate along isophotes, this paper puts forward a method of synthesizing texture via exemplars along isophotes. The priority of a given pixel is given by the confidence in the value of the pixel multiplied by the dot product of the isophote with the vector normal to the boundary of the fill region. Thus, strong isphotes are filled earlier than weak isophotes. There remain ways to optimize and improve this algorithm, including applying more efficient searching strategies like K-Nearest Neighbors and various different distance functions for comparing two regions of an image which show promise.**

I. Introduction

Inpainting refers to the process of filling in a region of an image such that the filled region appears reasonable to the human eye. The algorithm evaluated in this paper computes the most likely texture to replace the region to be filled.

At a fundamental level, an image I can be defined as a function of a vector containing the 2 dimensional coordinates of a pixel. In the inpainting problem, an image is assumed to be put through a system M that degrades it. The degraded region/s of the image can be described as Ω, such that $I = \Omega \cup \Phi$, where Φ is the source region of the image and Ω is the unknown or degraded region of the image. The observed degraded image F can be expressed as F = MI. [3]

Identify applicable funding agency here. If none, delete this.

The goal of inpainting is to estimate the degraded pixels of Ω from the known pixels in Φ. The key metric in doing so is the "reasonableness" of the estimated image to the human eye such that the estimated image appears as plausible as possible. In degraded images, there typically exist artifacts, including blurs, unconnected edges or inconsistent textures, all of which research of inpainting attempts to reduce and remove.

In past work, researchers have explored texture synthesis as a method of inpainting. These methods largely used repeated patterns from the source region of the image [1]. More specifically, the texture synthesis problem is best described as attempting to create a texture from a given sample such that the produced texture is larger than the sample but still plausible to the human eye. For the purposes of inpainting, these methods sample and copy colors from the source region of the image and determine how to fit patterns together to form a larger pattern. Fundamentally, this is based in replication, which is both its strength and weakness. This technique was effective in replicating consistent texture, but in real-world images, where linear, non repeating structures were present, they showed to be ineffective, producing either implausible images, excessive artifacts, or both [1].

Moreover, several algorithms filled regions using diffusion, a process that propagates higher order derivatives of pixel intensity along isophotes. These techniques are similar to physical processes like heat propagation through physical structures. Inpainting via diffusion extends local image structures from the border of the fill region towards the center [3]. As a method of preserving local structures, the propagation must follow image structures, parallel to the gradient of the image. While these methods showed significant promise in the restoration of im-

ages i.e. removing speckles, scratches, and overlaid text, when filling in large regions, they result in a blurring effect [2]. This becomes especially problematic when attempting to inpaint a large region of an image, resulting in a blurred mass that appears implausible to the human eye.

II. Methods

Beyond texture synthesis and diffusion alone, this paper [1] attempts to combine the most salient aspects of both approaches to produce an algorithm that produces more plausible results with fewer visual artifacts.

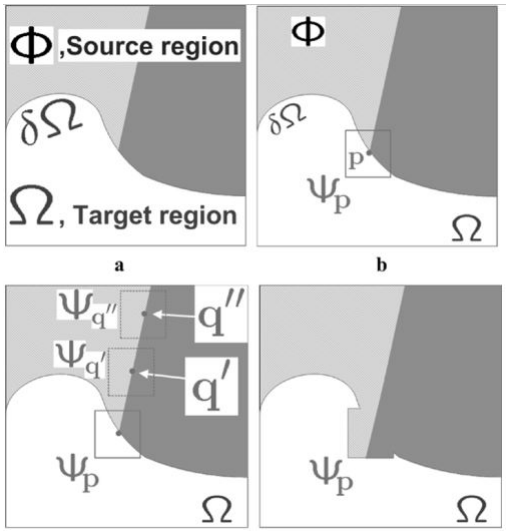

Fig. 1. (a) Notation of the image is shown, where Ω is the region to fill, Φ is the source region, and $\delta\Omega$ is the boundary of the fill region. (b) A point p is selected from $\delta\Omega$ to have the surrounding region Ψ_p filled. (c) Points q' and q'' from Φ and surrounding regions $\Psi_{q'}$ and $\Psi_{q''}$ are examined to determine the distance from Ψ_p. (d) The appropriate patch from Φ is identified and copied into Ψ_p. Image from [1].

As we have defined the fill region to be Ω, the boundary of that region can be defined as $\delta\Omega$. We can identify candidates to fill by their presence in $\delta\Omega$. As noted in [1], the order in which candidate points are filled is crucial to producing a plausible inpainting effect. This is demonstrated by Figure 2, where the filling of a concave target is shown from top to bottom, in an onion peel fashion, contrasted against the filling of the same target filled in order of the points with the strongest linear structures.

In practice, identifying $\delta\Omega$ proved difficult, but it was overcome by applying a laplacian filter to a mask. The mask indicated the region within the image to fill. A laplacian filter is a subpar edge

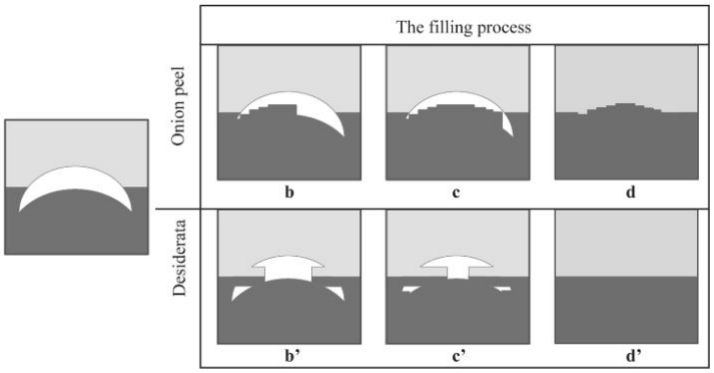

Fig. 2. The image on the left indicates the shape to be filled. (b-d) Artifacts are left behind when the fill order of Ω is not ordered based on priority. (b'-d') Minimal artifacts are left behind when Ω is filled in order of pixel priority. Image from [1].

detector, but met our expectations. The pixels of $\delta\Omega$ could be identified as being nonzero in the resulting filtered image. insert image of laplacian against mask

The filling priority of a given pixel in $\delta\Omega$ has two forces acting on it- the confidence the user has in the the surrounding pixels and the strength of the surrounding linear structures. Thus Criminisi et al has created a priority function P(p), where p is a given pixel to be filled. They define P(p) = C(p)D(p), where C(p) is the confidence in the pixel p and D(p) is a data term.

For our purposes, we can compute C(p) by taking the sum of confidence in the surrounding pixels and averaging them. The confidence is given by summing the confidence we have in Ψ_p and dividing by the number of pixels in Ψ_p. Confidence is initialized to 1 for all pixels in Φ and 0 for all pixels in Ω.

The data term is given by the magnitude of the dot product of the isophote and a unit vector normal to $\delta\Omega$, normalized by the maximum pixel value possible, typically 255. An isophote in the image can be identified via taking the gradient of the source image. The normal vector can be estimated by examining the area surrounding pixel p for other pixels in $\delta\Omega$ and applying the limit definition of a gradient (shown below), where, rather than a incrementally smaller timestep, we take an incrementally smaller pixel step.

$$lim_{\Delta x,y\to 0}\frac{f(x,y)-f(x+\Delta x,y+\Delta y)}{\Delta x\Delta y} \quad (1)$$

In practice, this was expedited by taking the gradient of the laplacian of the mask image. As

there were small pixel shifts between the location of $\delta\Omega$ and its gradient, a small window (3 pixels by 3 pixels) surrounding the pixel p were examined and the mean value of the gradient in the x and y directions were calculated and assumed to be the gradient at p.

Described above, exemplar-based texture synthesis can recreate plausible 2 dimensional structures by constructing a repeating pattern. This algorithm applies exemplar-based texture synthesis by identifying a patch Ψ_p to fill and determining the patch Ψ_q within Φ that minimizes the difference between the two patches. In our implementation, as well as the process described in [1], the difference was determined by the Sum of Squared Differences.

The patch Ψ_q is then copied into the patch Ψ_p. The boundary $\delta\Omega$ is updated with the new filled region, the confidence, data, and priorities of each pixel are recomputed, and the process continues until $\delta\Omega$ is empty.

III. Results

Two images were tested for inpainting with this algorithm. They are shown in figures 4 and 6. Figure x was constructed computationally and figure y was gathered using real world imaging. The results of the testing are shown in figures 3 and 5.

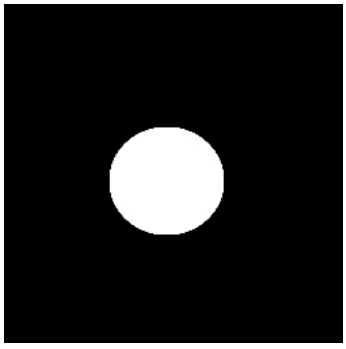

Fig. 3. Mask applied to the computationally constructed image.

As clearly indicated, the implementation did not match the results shown in [1]. This is likely due to a number of factors. First, the result shown in figure x+2 shows the characteristic visual artifact of the onion peel order of filling, which, in testing, was how the image ended up being filled according to the priority. This may have occurred because the normal vectors at the pixels bordering the strongest isophote were orthogonal to the isophote, resulting in a dot product equal to 0. This caused the priority

Fig. 4. The computationally constructed test image on the left and the result of the implementation of Criminisi et al's algorithm on the right. The region to remove is shown in figure 3

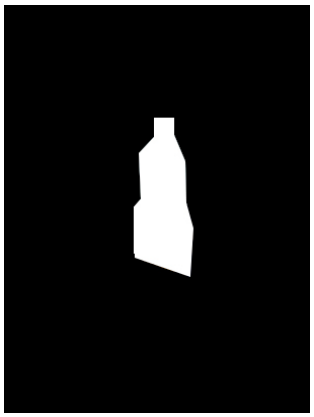

Fig. 5. Mask applied to the real world image.

of several pixels in $\delta\Omega$ to go to 0, negating the effect on the priority-based filling of the image.

Determining the difference between Ψ_p and a potential Ψ_q proved difficult as well. In the computationally constructed test image, a large portion of the image was black, corresponding to RGB values of 0. The squared sum of a Ψ_q from this region would be 0 or close to 0. If a large portion

Fig. 6. The real-world test image. The image inputted is shown on the left and the result of the implementation is shown on the right. The region to remove is shown in figure 5.

of Ψ_p had yet to be filled, the squared sum of Ψ_p would be close to 0 as well, resulting in a disproportionately high similarity between Ψ_p and the potential Ψ_q. This was somewhat resolved with a regularization scheme, but that is beyond the scope of the implementation, and there remains space for further investigation into higher quality methods of regularizing distance between patches.

IV. Discussion

While Ciminisi et al shows a great deal of success in their algorithm, there remain methods to improve its efficiency and robustness.

A key problem experienced while implementing their algorithm involved the selection of exemplars Ψ_q to place in patches Ψ_p. In the testing image shown in figure x, we have a region that is largely black. This equates to the pixel values being 0, 0, 0 for the values for red, green, and blue. As a result, the squared sum of this patch Ψ_q is 0. For a given patch Ψ_p with several of its pixels unfilled, its squared sum would be close to 0 as well, but the filled pixels may have values that do not match Ψ_q. As a result, the algorithm would copy Ψ_q in without the patch being a very close match to the human eye.

To solve this problem, a regularization scheme may be needed. In our implementation, such a scheme was applied. The squared sum of the candidate exemplar Ψ_q was regularized by the inverse proportion of the number of pixels in Ψ_p that were filled. This function accomplished its goal mathematically, but there may exist regularization functions that achieve better results in terms of plausibility to the human eye.

More than that, additional difference metrics between patches may be used. This algorithm essentially applies Euclidean distance between patches, but the Manhattan distance or the Hamming distance may be applied instead. For example, Ouattara et al put forward a method of methods of measuring distance based on Peak Noise to Signal Ratio and Mean Squared Error, as well as a novel weighting scheme for determining pixel priority, rather than simply multiplying confidence and data [**?**].

This algorithm has a significant time cost associated with running it. There are several processes running in $O(n^2)$, $O(n^3)$, or even $O(n^4)$ time, and as a single image dimension increases by one, the image scales by n times. This introduces squared time cost increase as the image increases in size, which significantly prolongs this process. There may be ways to make it more efficient by introducing Machine Learning methods or methods that are commonly applied in Machine Learning methods.

One might choose to apply the K-Nearest Neighbor algorithm to efficiently select the exemplar that minimizes the distance function. The K-Nearest Neighbor algorithm applies a KD Tree, a data structure that maps binary space to K-dimensional space, enabling $O(log(n))$ runtime when searching for a potential Ψ_q to copy into Ψ_p.

References

[1] Criminisi, A., et al. “Region Filling and Object Removal by Exemplar-Based Image Inpainting.” IEEE Transactions on Image Processing, no. 9, Institute of Electrical and Electronics Engineers (IEEE), Sept. 2004, pp. 1200–12. Crossref, doi:10.1109/tip.2004.833105.

[2] Shivaranjani, S., and R. Priyadharsini. “A Survey on Image Inpainting Techniques.” International Journal of Science and Research (IJSR), no. 11, International Journal of Science and Research, Nov. 2015, pp. 2414–17. Crossref, doi:10.21275/v4i11.nov151762.

[3] C. Guillemot and O. Le Meur, ”Image Inpainting : Overview and Recent Advances,” in IEEE Signal Processing Magazine, vol. 31, no. 1, pp. 127-144, Jan. 2014, doi: 10.1109/MSP.2013.2273004.

[4] Ouattara, Nouho, et al. “A New Image Inpainting Approach Based on Criminisi Algorithm.” International Journal of Advanced Computer Science and Applications, vol. 10, no. 6, 2019, https://doi.org/10.14569/ijacsa.2019.0100655.