CS766
Prof. Dyer

Object Removal Using Exemplar-Based Inpainting

Ye Hong

University of Wisconsin-Madison

Fall, 2004

Abstract

Two commonly used approaches to fill the gaps after objects are removed from digital graphics are: image inpainting and texture synthesis. Both approaches have its advantages and disadvantages. Image inpainting will cause blur, while texture synthesis will lead to loss of linear structures.

Criminisi, etc., combined the two approaches to a new algorithm that utilized the advantages of both approaches. Higher filling priority will be given to where linear structures are naturally extended into the gap. With this algorithm, the gap will be filled with non-blur textures, while at the same time preserve and extend the linear structure of the surrounding area.

In this report, I will first introduce the existing approaches to fill the gap. Then, I will explain Criminisi's algorithm in detail. The third part is the discussion of my design and implementation details. Results will be presented and analyzed. Lastly, I will spend the last part of the report to discuss about the possible future directions.

1. Introductions

Object removal from images is an image manipulation technique that has a long history. The purpose of removing objects varies from remove undesired object to improve the quality of the image, to “airbrushing” out political enemies from portraits of important events. Modern photographical manipulations, such as red eye removal from pictures, also utilized this technique. The process of removing objects from images starts with mask out the undesired object, making the area where the object previously occupies a gap. Then the gap will be filled using graphical techniques such as inpainting. Among the graphical techniques that are used to fill the gap after object removal, two most commonly used are: image inpainting and texture synthesis.

Fixing images using inpainting has a long history. Most notably, during the Renaissance, many medieval artworks had been brought “up to date”. Missing or damaged parts in the paintings were reconstructed in a way that not detectable from human eyes. Structures and textures around the gap were carefully extended into the missing area. The results would look natural enough that observers without prior knowledge of the original image will not notice the gaps. Bertalmio, etc.[2]’s algorithm imitates the traditional inpainting processes, such as determine the area to be corrected, exam the boundary of the region to be filled, and continuing lines of similar color. A series of differential equations are used to imitate these processes. The most important equation is to calculate the isophote (direction and intensity) of a pixel. Each pixel is modified by adding its current intensity to and updated intensity times a delta factor. The updated intensity consists of a change of smoothness estimation projected along the direction of shortest change. Smoothness is approximated by a discrete Laplacian. The direction of shortest change is defined as the vector perpendicular to the gradient, and the dot product of this vector with the vector consisting of the x and y changes in the Laplacian, is multiplied by a slope-limited norm of the gradient of numerical stability.

Texture synthesis grows a new image outward from an initial seed. Efros and Leung [3] proposed a texture synthesis approach by non-parametric sampling. A Markov random field model is assumed. Before a pixel is synthesized, its neighbors are sampled. Then the whole image is queried to find out a pixel (source pixel) with similar neighbors. At this point, the source pixel is copied to the pixel to be synthesized. The biggest draw back of this algorithm is that generating every single new pixel needs to sample the whole picture, and thus not very efficient. Efros later went on to address this issue by modifying the algorithm to perform the sampling, search and synthesis in a patch. Each patch has an overlap with the next one, and they are quilted together. [4] This new algorithm improved performance greatly.

2. Criminisi's Algorithm

Both image inpainting and texture synthesis have their strengths and weaknesses. Image inpainting extends linear structure into the gap by utilizing the isophote information of the boundary pixels. The linear structures will be naturally extended into the gap. However, since the extension actually using the diffusion techniques, artifacts such as blur could be introduced. On the other hand, texture synthesis copies the pixels from existing parts of the image, thus avoids the blur. The shortcoming of texture synthesis is that it focuses on the whole image space, without giving higher priority to the linear structures around the boundary of the gap. As a result, the linear structures will not be naturally extended into the gap. The result would likely have distorted lines, and noticeable differences between the gap and its surrounding area would be expected.

One interesting observation is that even though image inpainting and texture synthesis seem to differ radically, yet they might actually complement each other. If we could combine the advantages of both approaches, we would get a clear gap filling that is the natural extension from the surrounding area. Criminisi, etc. proposed a new algorithm that does exactly this. [1]

Criminisi, etc. use the sampling concept from Efros' approach. The improvement over Efros' is that the new approach takes isophote into consideration, and gave higher priority to those "interesting points" on the boundary of the gap. Those interesting points are parts of linear structures, and thus should be extended into the gap in order to obtain a naturally look. To identify those interesting points, Criminisi gives a priority value to all the pixels on the boundary of the gap. The "interesting points" will get higher priorities according to the algorithm, and thus the linear structures would be extended first.

For each pixel on the boundary, a patch is constructed with that pixel at the center. The patch's priority is the product of two elements: a confidence term C(p), and a data term D(p). C(p) describes how many pixels are there in the patch. It is obvious that with more pixels in the patch, we would have a better confidence that a success target patch would be selected. D(p) describes how strong the isophote is hitting the boundary. This term boosts the priority of a patch that an isophote “flows” into. D(p) is especially important, since it encourages linear structures to be synthesized first, and thus propagated securely into the target region.

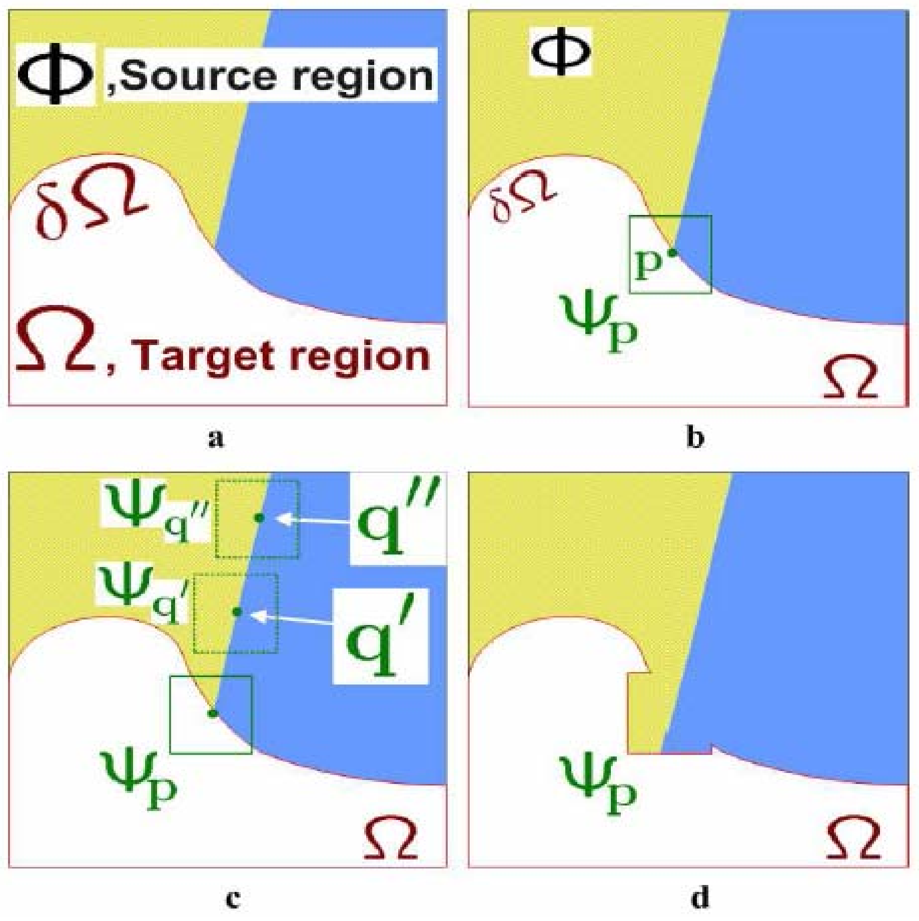


Figure 1. Structure propagation by exemplar-based texture synthesis.

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 center 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.

3. Implementation Details

++. The program will take in two input images: the original image and the Photoshop mask that would mask out the object.

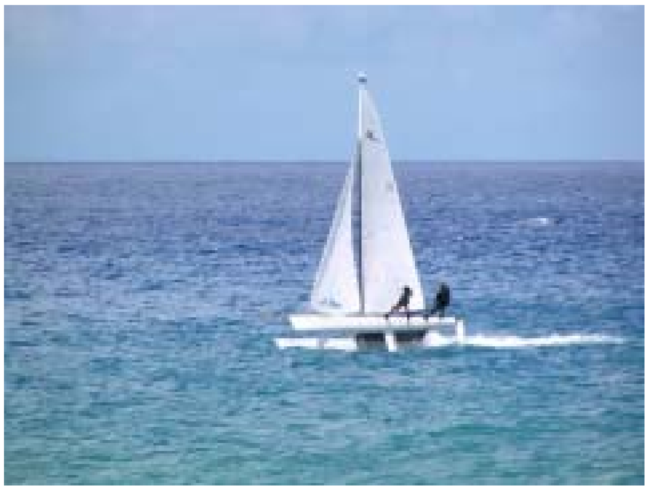
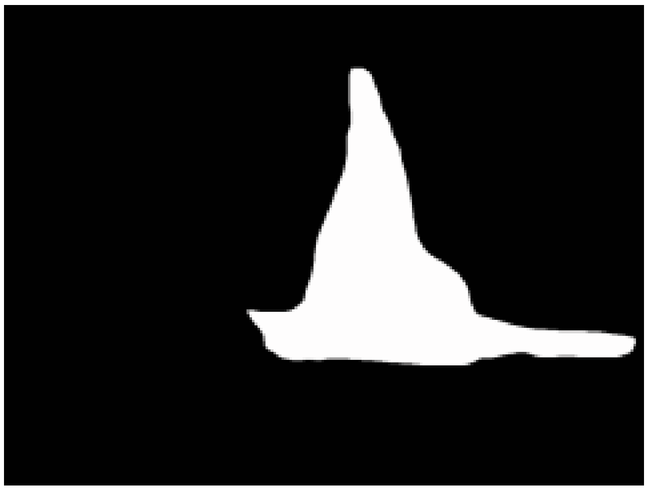
After read in the mask, I marked the target region that will be filled. Also, the contour (the boundary between the gap and the surrounding area) is defined as a collection of pixels that has a neighboring pixel in the target region. I maintain a list of contour points in the form of a vector. For each pixel in the contour, I built a patch with a given size. Then, I apply Criminisi's algorithm to find out the target patch that has the highest priority.

Then, I performed a search to find a patch in the source area that is the most similar to the target patch. In my implementation, I calculate the color distance between the non-empty patch pixels at the same position. Sum of Square Difference (SSD) is used to calculate each color channel's difference, and then use SSD to sum up the overall color distance between the pixels.

Once the best-fit patch has been found, I copy the color 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.

After filling the patch, I update 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 Figure 1. (d).

I keep select patches whose center point is on the contour to be filled. After the filling, I would update the contour list. Eventually, the whole target region will be entirely filled, and the contour list would be empty. At this point, I have my result picture.

Original Image **Mask**

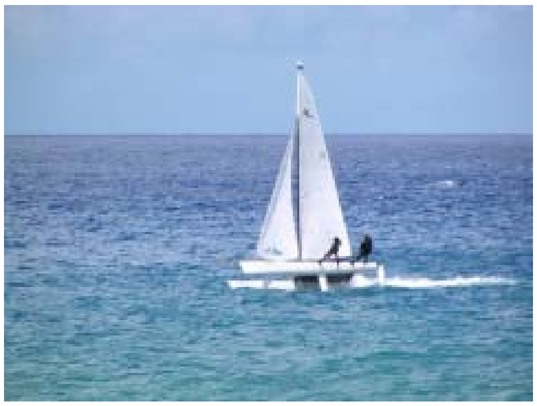
Figure 2. Input Images.

I implement the Criminisi algorithm with standard C++. The compilation and execution of the program are in Cygwin environment under Windows XP. The compiler is gcc version 3.3.3 (cygwin special).

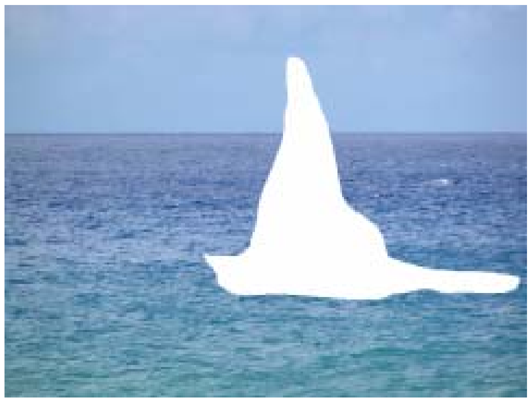
4. Results and Discussions

In my results, I had experimented with a few parameter of the Criminisi algorithm.

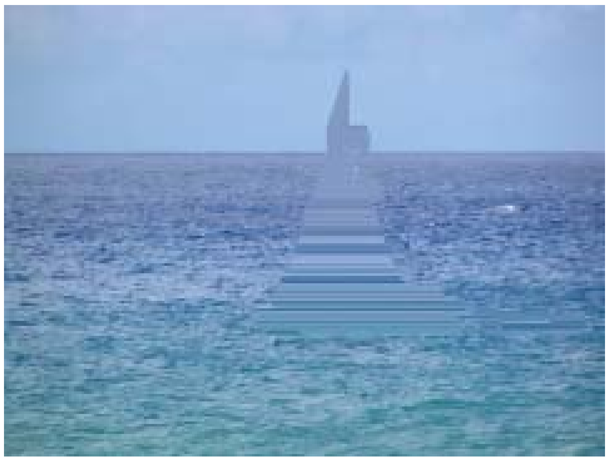
One important parameter of the algorithm is the size of the patch. With bigger patch size, the filling rate is high, thus the program runs faster. However, there's more important implications on choosing the right patch size.



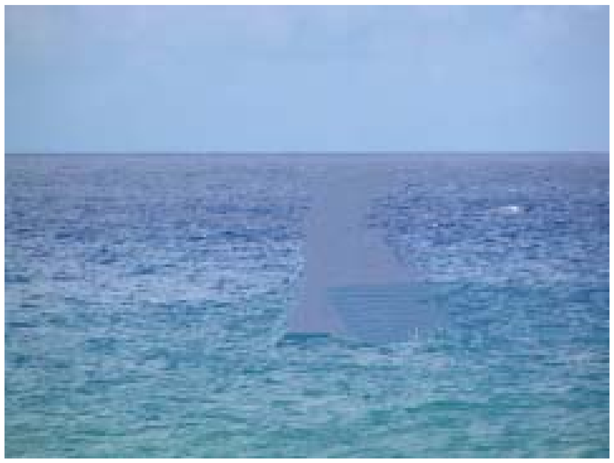
Original



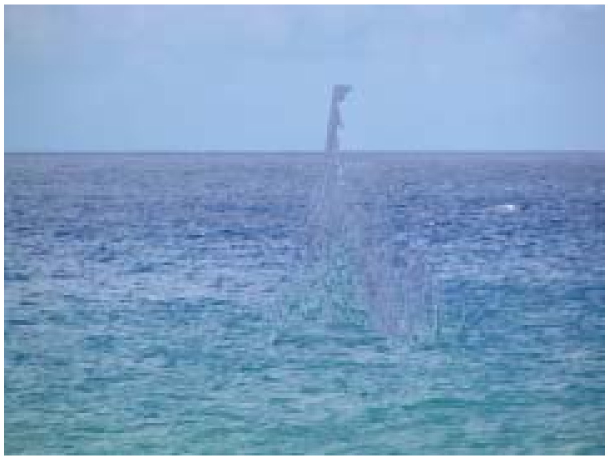
With Object Cut Out



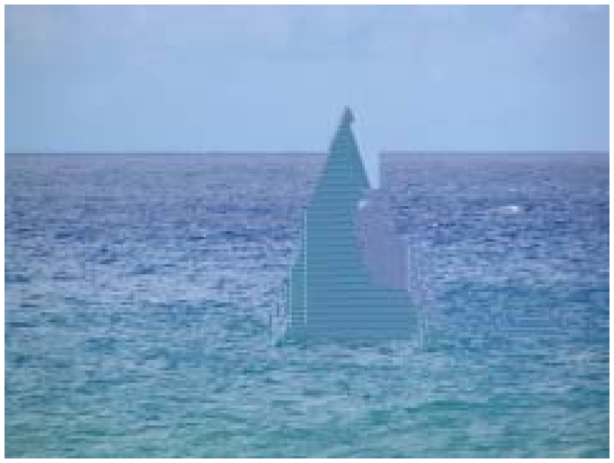
Patch Size = 5x5



Patch Size = 7x7



Patch size = 9x9

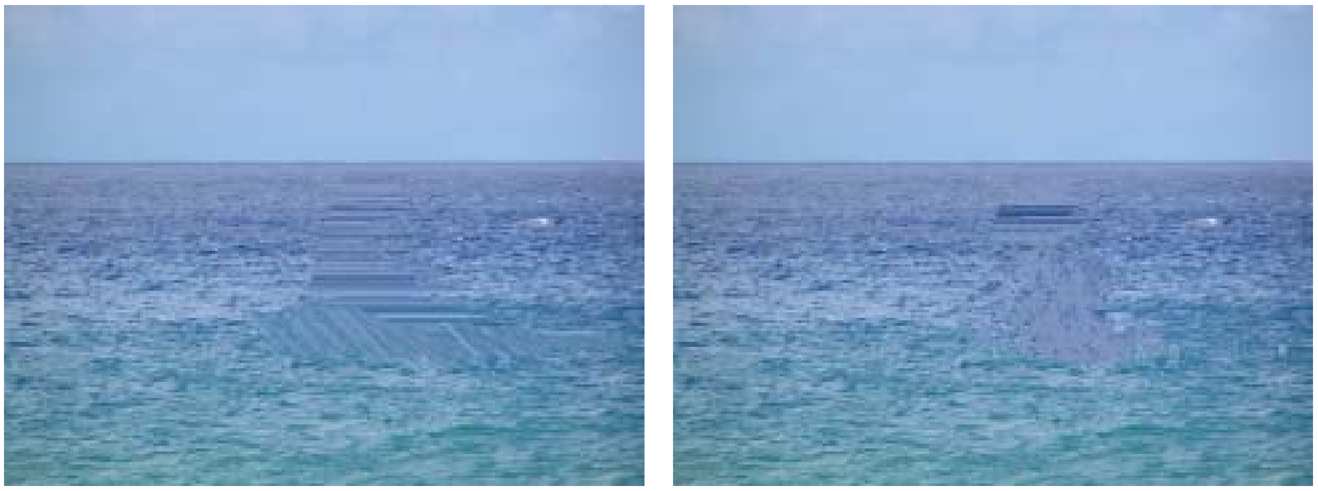


Patch Size = 11x11

Figure 3. Results with different patch size

Criminisi stated in his paper that the patch should be slightly larger than the largest distinguishable texture element. He gave 9x9 as default in his paper. As Figure 3 shows, 9x9 is not the best choice in the sample image used in this report. I have 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. 5x5 and 11x11 look very bad.

The conclusion I drew from the patch size experience is that most of the texture elements is around 7 and 9. The reason that other sizes fail might because of the bad samplings.



Isophote off **Isophote on**

Figure 4. The effect of Isophote

The next experiment I did is to see how the isophote would affect the results. Since 7x7 is the best patch size in the previous experiment, I use it as the default in this experiment. The right image is the result of using isophote, it blends with the surrounding much better, the texture structures in the surrounding area are well preserved. In contrast, the direction of textures on the left image looks very unnatural. There are still some artifacts on the right image, but it is more likely an implementation error.

5. Future Works

Future directions could be as such:

In addition to preserve the linear structures well in the target region, the way to give curve structures a naturally-look extension would worth some investigation.

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 boost the performance, so that this algorithm could be used to video applications.

The quality of the result images is not as good as expected. I will work to improve it in the future.

References:

[1] A. Criminisi, P. Perez, K. Toyama.
Region filling and object removal by exemplar-based inpainting.
In *2004 IEEE Transactions on Image Processing 9 1200-1212.*

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester.
Image inpainting. In *Proc. ACM Conf. Comp. Graphics (SIGGRAPH)*, pp. 417–424, New Orleans, LU, Jul 2000.

[3] A. Efros and T. Leung.
Texture synthesis by non-parametric sampling.
In *Proc. ICCV*, pp. 1033–1038, Kerkyra, Greece, Sep 1999.

[4] A. Efros and W.T.Freeman.
Image quilting for texture synthesis and transfer.
In *Proc. ACM Conf. Comp. Graphics(SIGGRAPH)*, pp. 341-346, Eugene Fiume, Aug 2001.