IMAGE INPAINTING AND TEXTURE SYNTHESIS: TWO METHODS FOR HOLE FILLING IN IMAGES

Siddharth Jain

morpheus@eecs.berkeley.edu
Department of Electrical Engineering and Computer Science
University of California, Berkeley

ABSTRACT

This paper presents a discussion and comparison of two algorithms for Hole Filling in images: given an image with regions of missing RGB values (*holes*), determine the RGB values or fill the holes from the information available from the rest of the image. The hole filling problem occurs commonly in image processing/computer vision when unwanted objects have to be removed from images or some sort of disocclusion has to be performed. The first algorithm we discuss for hole filling is due to [1], referred to as the image *inpainting* algorithm that attempts to fill the hole by stretching the geometric and photometric information available in a thin band around the hole. The second algorithm is a copy-paste method based on the idea of texture synthesis in [2]. Experimental results and analysis are provided.

1. INTRODUCTION

Consider the problem of hole filling in images: given an image with regions of missing RGB values (*holes*), determine the RGB values from the information available from the rest of the image. An example where this problem arises is the removal of unwanted objects from images or handling occlusion in images. Figure 2(a) shows an image in which the building façade has been occluded by some foreground objects marked as white in the image and we would like to reconstruct what is hidden behind the foreground objects based on the information available from the rest of the image.

We discuss and compare two methodologies for hole filling in images — the first one is due to [1] referred to as image *inpainting*; the idea is to take a band around the hole and to fill it using the geometric and photometric information contained in this band. The second approach is based on synthesizing artificial texture in the hole. Before we begin a discussion of hole filling algorithms, it is important to realize that the hole filling problem is an intrinsically ill-posed problem. Given a hole that is to be filled, there is no clear and unique answer as to what the hole filled image should look like. More importantly, there is no well defined metric that tells whether one hole filled result is better than another. The criterion that we have used to judge the performance of a hole filling algorithm is whether the hole filled image appears visually pleasing to a human observer or not.

The organization of the paper is as follows: in section 2 we introduce the image inpainting algorithm. Section 3 introduces the copy-paste method. Section 4 provides some implementation details related to image inpainting. Section 5 discusses and compares the two algorithms and presents experimental results. Finally we end with conclusions in section 6.

2. THE IMAGE INPAINTING ALGORITHM

This algorithm was introduced by [1] inspired by the works of [3, 4] and has been extended in [5, 6, 7, 8]. In order to fill a hole Ω, a thin band is taken around the hole and the algorithm attempts to fill the hole using the geometric and photometric information contained in the band. This is done by using a variational continuation framework and attempting to continue the level sets of the image inside the hole by minimizing an energy functional. Specifically as per [1], every pixel (x, y) in the hole, is updated iteratively as follows:

$$I^{n+1}(x,y) = I^n(x,y) + \Delta t I_t^n(x,y) \forall (x,y) \in \Omega \quad (1)$$

where Δt is a time step set equal to 0.1 and $I_t^n(x,y)$ is the update at pixel (x,y) given by

$$I_t(x,y) = \left(\delta L(x,y) \cdot \frac{N(\vec{x},y)}{|N(\vec{x},y)|} \right) |\nabla I(x,y)| \quad (2)$$

$$\delta L(x,y) = (L(x+1,y) - L(x-1,y), L(x,y+1) - L(x,y-1)) \quad (3)$$

where L is the laplacian given by

$$L(x,y) = I_{xx}(x,y) + I_{yy}(x,y) \quad (4)$$

and

$$N(\vec{x},y) = \nabla^{\perp} I(x,y) \quad (5)$$

is the normal to the gradient at (x,y).

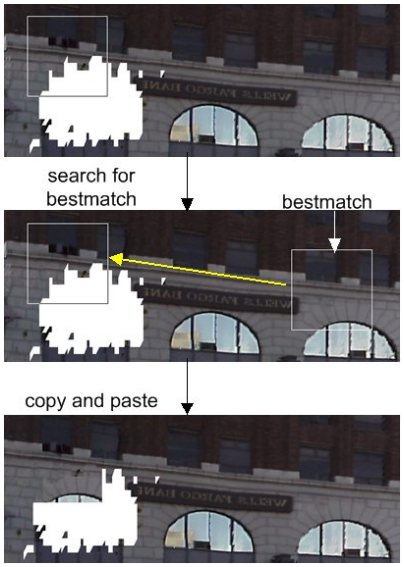

Fig. 1. Illustrating the Copy-Paste method

After every A = 15 steps of inpainting as described by above equations, B = 2 steps of anisotropic diffusion are applied to mitigate the effects of noise while preserving the edges.

3. THE COPY PASTE METHOD

This method of hole filling is based on the idea of synthesizing artificial texture in the holes. For an early paper on hole filling using texture synthesis see [9]. The copy-paste method we describe here is conceptually a very simple method based on the idea of texture synthesis in [2]. The holes are filled by copying and pasting suitable blocks from other parts of the image. The method is illustrated in Figure 1. For recent similar and independent work see [10].

The image is scanned pixel by pixel in raster scan order and pixels at the boundary of holes are stored in an array to be processed. A window w is taken centered at a hole pixel p and the image is searched for a window bestmatch(w) which (a) has the same size as w (b) lies in a search region w_s which typically is a large window having same center as w (c) does not contain more than 10% hole pixels and (d) matches best with w. If the difference between w and bestmatch(w) is below a threshold, the bestmatch is classified as a good match to w and hole pixels of w are replaced with corresponding pixels in bestmatch(w). For the method to work well we need a suitable metric that accurately measures the perceptual difference between two windows, an efficient search process that finds the bestmatch of a window w fast, a decision rule that classifies whether the bestmatch found is a good match or not, and a strategy of dealing with cases when the bestmatch of a window w is not a good match. In our algorithm the difference between two windows consists of two components: (a) the sum of color differences of corresponding pixels in the two windows, and (b) the number of outliers for the pair of windows. These components are weighted appropriately to compute the resulting difference. An efficient search is performed by constructing a hierarchy of Gaussian Pyramids [11], and performing an exhaustive search at the coarsest level to find a few good matches, which are then successively refined at finer levels of the hierarchy. We use an adaptive window size to find matching blocks in the image from where appropriate texture can be copied and pasted over the hole. In cases when no good match is found, the hole pixels are filled by averaging the known neighbors if the pixel variance of the neighbors is low; otherwise the colors of hole pixels are set equal to the value of randomly chosen neighbors. More details can be found in [12].

4. INPAINTING — IMPLEMENTATION DETAILS

We have tried to implement a reasonably efficient version of the inpainting algorithm as described in [13]. Following points are worth mentioning:

- For an image define

$$\gamma = \frac{\text{number of hole boundary pixels}}{\text{total number of hole pixels}} \tag{6}$$

  In this paper we will associate the notion of thickness of a hole to the value of γ for an image in the following way: thick holes imply small values of γ and vice versa.

- If the holes are thick, Gaussian Pyramids of the image are created and the algorithm works from coarser to finer levels. Specifically the image is convolved with a Gaussian and subsampled repeatedly while (a) $\gamma < 0.4$, and (b) number of hole pixels $>= 512$, and (c) number of subsampling operations performed < 4.

- **Initialization**: The pixel values in the holes are initialized as follows: Consider a 3×3 window centered at an unknown pixel. Perhaps the simplest way to fill the pixel is to average the values of the known pixels in the 3×3 window and assign the average value to the unknown pixel. But if the values of neighbors differ significantly then averaging can lead to appearance of spurious colors and blurring. In case of high pixel variance, instead of averaging, the unknown pixel can be assigned the value of a randomly chosen neighbor. We call this method of interpolation as SimpleInterpolation. The holes are initialized using SimpleInterpolation starting from the boundary pixels and propagating inwards. This is the same technique we use

in the copy-paste method to fill holes when no good matches are found in the image. In fact if the hole pixels are filled by averaging the neighbors starting from the periphery of the hole and propagating inwards, then the result obtained approximates the solution of the Laplace Equation in the hole subject to known boundary values, the so called Dirichlet problem in classical physics. This can be proved by using the mean value theorem [14].

5. INPAINTING VS. COPY-PASTE

We tested the inpainting and copy-paste method on a dataset of 8 images. Two test cases comprise of input images that are taken from earlier papers [1] and can be used to verify our implementation of the inpainting algorithm. All the computation is done on an Intel Pentium processor running at 2.0 GHz and operating on Windows OS. A single set of parameters is used in all the computations. Due to space restrictions we only show the images for two test cases in this paper in Figure 2-3. The results for the other test cases can be seen at `http://www.awargi.org/holefilling/index.html`. Table 1 shows approximate processing time.

It is found that the inpainting methods are appropriate for filling thin holes in non-textured regions but suffer from the limitation of *local inpainting* i.e. image inpainting does not rely on global feature or pattern recognition [7]. Also it must be remarked that when holes are thin, other simpler methods like linear interpolation of surrounding values or SimpleInterpolation introduced in this paper give results comparable to that obtained with inpainting and at a fraction of the computational cost. The copy-paste method on the other hand performs better on thicker holes and is more suited for filling texture holes in images as evidenced in the figures and table 1. Following salient features of the copy-paste method emerge:

- The method works well on holes in repetitive patterns, textures and smooth regions.
- The method is able to detect and complete straight edges and linear features satisfactorily because it is able to take a small sample of a line and replicate it to extend the line. Note this for example in figure 3.
- The method does not suffer from the limitation of local inpainting.
- The method is capable of filling holes in an iterative fashion that are larger than the matching window itself.
- There is an inherent assumption that there exists some region in the image which can be copied and pasted over a hole — this actually turns out to be true in many natural images. Nevertheless there should be some strategy of dealing with cases when there is no good match that can be copied and pasted.

Image	γ	Inpainting	Copy-Paste
Graffiti1	0.8347	0.4	0.4
Graffiti2	0.7372	0.3	0.3
Campanill	0.1898	0.3	0.1
Atlas42	0.1226	>2	0.15
Atlas54	0.1090	>2	1.86
WellsFargo	0.0783	>2	0.07
Vegas3	0.0425	1.55	0.08
Pier39	0.0386	1.36	0.02

Table 1. Table showing approximate processing time in hours for the test cases in the paper.

Finally, in its basic form (without making a Gaussian Pyramid hierarchy), the runtime of the inpainting algorithm is linear in the number of hole pixels in the image. The runtime of the copy-paste method is not so easy to analyze and depends on various factors:

- The size of the search region in which the matching block is searched.
- If the algorithm is able to find large matching blocks it fills the holes much more rapidly than in the case when matching blocks are small sized or when there are no matching blocks at all.

Nevertheless, usually the copy-paste method fills the holes faster than the inpainting algorithm as evidenced from the results.

6. CONCLUSION

We presented a discussion of two methodologies for hole filling — inpainting and texture synthesis. Inpainting performs better when γ is large such as $\gamma > 0.6$ and suffers from the limitation of local inpainting which limits its usefulness for filling texture holes in images. Texture synthesis based hole filling such as the copy-paste method described here is more suited for filling texture holes and holes that have lower values of γ. An interesting paper that combines the two approaches is [15] in which the authors fill holes due to lost blocks in wireless video transmission; the algorithm decides whether the hole is in a textured region or not using the test described in [16]. If the hole lies in a non-textured region it is filled using inpainting; otherwise it is filled using texture synthesis.

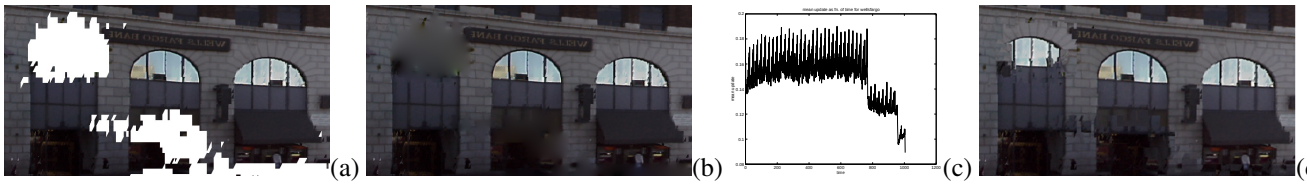

Fig. 2. WellsFargo. $\gamma = 0.0783$. Original image size 667×349 (a) input image (b) result of inpainting after more than 2 hours (c) mean update given by equation (2) plotted as a fn. of time. The jumps in the plot occur when the algorithm switches from one level to another in the Gaussian Pyramid hierarchy. (d) result of copy-paste. time taken = 0.07 hours.

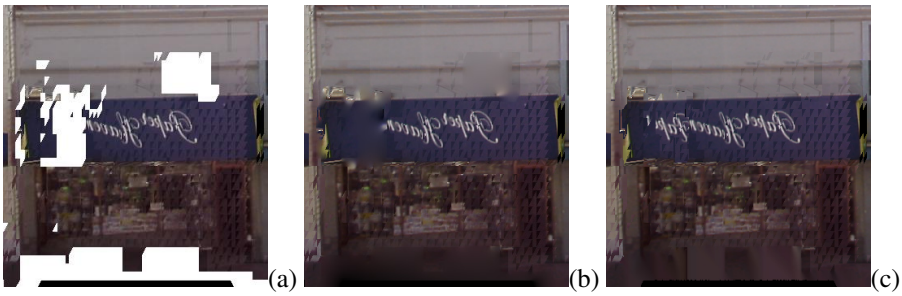

Fig. 3. atlas42. $\gamma = 0.1226$. Original image size 326×347 (a) input image (b) result of inpainting after more than 2 hours (c) result of copy-paste. time taken = 0.15 hours.

7. ACKNOWLEDGEMENTS

This work was sponsored by Army Research Office under contract DAAD19-00-1-0352.

8. REFERENCES

[1] Marcelo Bertalmio, Guillermo Sapiro, Vicent Caselles, and Coloma Ballester, "Image inpainting," in *SIGGRAPH 2000, Computer Graphics Proceedings*, Kurt Akeley, Ed. 2000, pp. 417–424, ACM Press / ACM SIGGRAPH / Addison Wesley Longman.

[2] Alexei A. Efros and William T. Freeman, "Image quilting for texture synthesis and transfer," in *SIGGRAPH 2001, Computer Graphics Proceedings*, Eugene Fiume, Ed. 2001, pp. 341–346, ACM Press / ACM SIGGRAPH.

[3] M. Nitzberg, D. Mumford, and T. Shiota, *Filtering, Segmentation and Depth*, Springer-Verlag, Berlin, 1993.

[4] S. Masnou and J.M. Morel, "Level-lines based disocclusion," in *Fifth IEEE International Conference on Image Processing*, 1998, pp. 259–263.

[5] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling in by joint interpolation of vector fields and gray levels," *IEEE Trans. Image Processing*, pp. 1200–1211, August 2001.

[6] C. Ballester, V. Caselles, J. Verdera, M. Bertalmio, and G. Sapiro, "A variational model for filling-in gray level and color images," in *Proc. Eighth IEEE International Conference on Computer Vision*, 2001, vol. 1, pp. 10–16.

[7] Tony Chan and Jianhong Shen, "Mathematical models for local nontexture inpaintings," *SIAM Journal on Applied Mathematics*, vol. 62, no. 3, pp. 1019–1043, 2001.

[8] Tony Chan and Jianhong Shen, "Nontexture inpainting by curvature-driven diffusions (cdd)," *J. Visual Comm. Image Rep.*, vol. 12, no. 4, pp. 436–449, 2001.

[9] H. Igehy and L. Pereira, "Image replacement through texture synthesis," in *Proc. Fourth IEEE International Conference on Image Processing*, 1997, pp. 186–189.

[10] A. Criminisi, P. Perez, and K. Toyama, "Object removal by exemplar-based inpainting," in *IEEE CVPR*, 2003, pp. 721–728.

[11] D.A. Forsyth and J. Ponce, *Computer Vision — A Modern Approach*, chapter 7, Prentice-Hall, 2002.

[12] Siddharth Jain, "Hole filling algorithms for images with applications to automatic generation of 3D building façades," M.S. thesis, University of California, Berkeley, May 2003.

[13] Marcelo Bertalmio, *Processing of Flat and Non-Flat Image Information on Arbitrary Manifolds using Partial Differential Equations*, Ph.D. thesis, University of Minnesota, 2001.

[14] D. J. Griffiths, *Introduction to Electrodynamics*, chapter 3, pp. 112–115, Prentice-Hall, second edition, 1989.

[15] S. Rane, G. Sapiro, and M. Bertalmio, "Structure and texture filling-in of missing image blocks in wireless transmission and compression," in *Proc. Ninth IEEE International Conference on Image Processing*, 2002, vol. 1, pp. 317–320.

[16] K. Karu, A.K. Jain, and R.M. Bolle, "Is there any texture in the image?," in *Proc. 13th International Conference on Pattern Recognition*, 1996, vol. 2, pp. 770–774.