Digital Image Restoration Techniques And Automation

Ninad Pradhan
Clemson University
npradha@clemson.edu
a course project for ECE847 (Fall 2004)

Abstract

The restoration of digital images can be carried out using two salient approaches, image inpainting and constrained texture synthesis. Each of these have spawned many algorithms to make the process more optimal, automated and accurate. This paper gives an overview of the current body of work on the subject, and discusses the future trends in this relatively new research field. It also proposes a procedure which will further increase the level of automation in image restoration, and move it closer to being a regular feature in photo-editing software.

Keywords: image restoration, inpainting, texture synthesis, multiresolution techniques

1. Introduction

Image restoration or 'inpainting' is a practice that predates the age of computers. It is a technique used by art museum craftsmen to fill in parts of a painting which have decayed or been damaged over the course of many years. It is called inpainting as a literal terms for the process of painting in holes or cracks in an artwork. Digital image restoration has received attention over the last few years, because of the many applications that is has in image processing, including removal of scratches, objects, text or logos from digital images, while still retaining the consistency of the image in those areas which have been restored or retouched.

The problem in inpainting is posed as follows: given an area to be inpainted, filling in the missing areas or modifying the damaged ones in a non-detectable way for an observer not familiar with the original images [4]. Texture synthesis accepts a given sample texture and creates an output image which can have arbitrary dimensions, but which retains the texture properties derived from the original sample [15].

Both these approaches have different direct applications: where inpainting is used explicitly for filling holes in an image, texture synthesis draws from natural or artificial textures to create a textured pattern, which finds extensive applications in graphics and 3-D animation [17]. However, they are basically methods used to synthesize a pixel given some information about another set of pixels. We can think of the texture synthesis problem, which requires an input texture, as reducing to the inpainting problem if we assume that the sample or input texture which it attempts to replicate can be found in the same image where the region to be synthesized lies.

Hence we see that, for two dimensional digital images, one can use both texture synthesis and inpainting approaches to restore the image. This is the motivation behind studying work done in these two fields as part of the same paper. The two approaches can be collectively referred to as hole filling approaches, since they do try and remove unwanted features or 'holes' in a digital image.

The common requirement for hole filling algorithms is that the region to be filled has to be defined by the user. This is because the definition of a hole or distortion in the image is largely perceptual. It is impossible to mathematically define what a hole in an image is. A fully automatic hole filling tool may not just be unfeasible but also undesirable, because in some cases, it is likely to remove features of the image which are actually desired. The user should have control over the selection of the region, after which the filling is entirely automatic across the spectrum of methods to fill texture or inpaint.

Some attempts have been made at automatic hole filling, notably ones by Shih et al [9], who use a GUI tool which automatically fills an image, but later allows user to undo corrections to certain parts of the image. This technique uses the power to noise ratio in a pixel window and some global image statistics to determine if the inpainting operation should be carried out. I have proposed a method similar to theirs in its use of multiple resolution windows to detect a hole in the image, but which diverges in the details and hands over control of the inpainting to the user before the actual inpainting operation is performed by software.

Thus, this paper collates some of the defining work in inpainting and texture synthesis, which is of relevance to 2D image restoration, and finally proposes an approach to take the process towards more user-friendliness and automation.

2. Inpainting

There are two schools of thought on approaching an inpainting problem: making use of information about the local neighbourhood of a pixel only, and making use of global statistics to aid the local statistics.

2.1. Local inpainting

Bertalmio et al [4] define the problem of local inpainting as, given a region to be inpainted Ω and its boundary ∂Ω, to synthesize pixel values from the boundary inwards, using neighbourhood pixel information to continue the inpainting process.

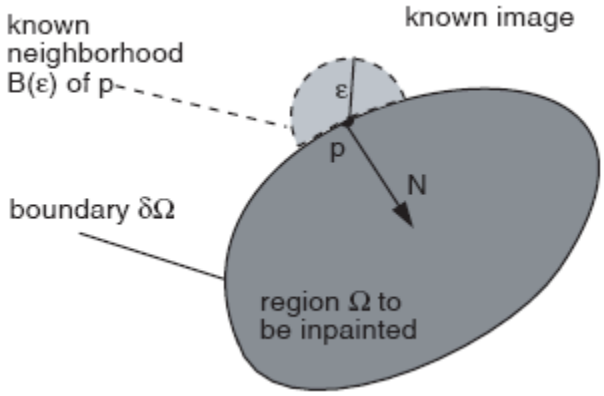

Fig 1: Local inpainting [5]

These methods assume a prior information about the probability distribution of the relation between a pixel value and its neighbourhood, which will help fill in a pixel lying on the hole boundary. Also, other properties such as high smoothness, low total variation or low curvature are assumed to create the framework on which the actual algorithm runs.

There are two significant approaches to local information based hole filling of images:

- Based on Partial Differential Equations
- Based on convolution

2.1.1. PDE based methods

The seminal work of Bertalmio et al [4] illustrates the PDE based method aptly. They provide an elegant mathematical treatment of the inpainting problem.

They define isophotes, or lines of equal gray value, as having to propagate inside the image boundary for the part of the image to be inpainted. This process is similar to estimating the optical flow in the neighbourhood of the boundary, and following these lines of flow, or isotopes, to the interior of the hole. Smoothness of flow is achieved by use of discrete 2D Laplacian, and the variation thus obtained is projected into the isophotes direction.

Authors also utilize anisotropic diffusion every few iterations [8] to minimize the effect of noise. Anisotriopic diffusion helps connect lines at the boundary of the inpainted region.

Maintaining the isophote direction is a major sub-problem for PDE based methods. The Laplacian is used by Bertalmio et al. The Curvature-Driven Diffusion (CDD) model takes into account geometric information of isophotes when defining the "strength" of the diffusion process, thus allowing the inpainting to proceed over larger areas.

PDE based methods are mathematically sound, do not require any user intervention, once the region to be inpainted has been selected and no assumptions on the topology of the region to be inpainted, or on the simplicity of the image, are made.

Fig 2: Results from Bertalmio et al [4]

However, there is a significant price to pay for these advantages [5]. These methods are extremely slow for practical applications, practical implementation details are not easily available, and require solution of nontrivial iterative equations (which also contributes to making them computationally expensive, and slowing them down).

2.1.2. Convolution based methods

These methods [6] make use of psychological findings that the human brain is more sensitive to edge information than information which gives details of a texture in the image, and reason that human visual system can tolerate some amount of blurring in areas not associated with high contrast edges.

Their second tenet is the use of sampling theorem, which imposes a fundamental limit on the quality of the information, in our case, scale of the texture, that can be restored by any automatic inpainting model.

These methods use the above principles to come up with a simple diffusion algorithm. This algorithm initializes the hole to zero colour information, and repeatedly convolves the image with a kernel which is rotationally symmetric.

The positives are that the algorithm is really fast and works well for images which do not have many high contrast edges or high frequency components (e.g. natural textures).

Fig 3: Results from Oliveira et al [6]

Since there is no 'intelligence' associated with the convolution, the algorithm gives poor results at high contrast edges. A user defined convolution barrier has to be put in place to retain edge information, which reduces the degree of automation. Images with large amounts of high frequency texture also exhibit errors when the convolution algorithm is run over them.

2.1.3. The FMM (Fast Marching Method)-based algorithm

This algorithm [5] can be looked at as the PDE based approach without the computational overheads. It is considerably faster than the PDE based algorithms, simple to implement, and produces results which are comparable to those given by the PDE method. It is also better positioned technically than the brute force convolution method.

The salient features of the algorithm are that an estimator is used for image smoothness (simplifies computation of flow), and smoothness is estimated as a weighted average over a known image neighborhood of the pixel.

The FMM is responsible for ensuring that the pixels closest to the known region get painted first, which is similar to the manner in which actual inpainting is carried out. FMM maintains a narrow band which separates known pixels from unknown, and also specifies which pixel to inpaint next.

A directional component is used to weight the contribution of the pixel closest to target pixel more than those further away.

2.2. Global inpainting

Inpainting based on global statistics [8] is a fledgling concept but worth a mention here because it highlights a potential weakness in local inpainting. Local algorithms give identical completions when the immediate boundary of the hole is identical. Human visual system takes more global information into account. Global statistics are similarly important in painting restoration. The major drawback with respect to this paper is that the algorithm does not perform well with natural textures as seen in photographs.

It captures the "look" of a training image in a probability distribution over images, using image histograms. Probability of an image is defined by means of a small number of sufficient statistics or features, each of which can be evaluated at an arbitrary location in the image.

Optimization is done using loopy Belief Propagation, a method used to pick a value from the probability distribution of the image given the hole boundary.

The drawbacks in the current implementation of global statistics-based inpainting are that the probability model does not capture texture very well, and the optimization method yields results which are not as good as expected.

We now look at the other approach, namely texture synthesis.

3. Texture Synthesis

3.1. Problem definition

Efros and Leung [15] have formulated the problem of texture synthesis as follows:

'Let us define texture as some visual pattern on an infinite 2-D plane which, at some scale, has a stationary distribution. Given a finite sample from some texture (an image), the goal is to synthesize other samples from the same texture. Since a given texture sample could have been drawn from an infinite number of different textures, it is assumed that the texture sample is large enough to capture the stationarity of the texture'. A texture is said to be stationary when, if we move a window over an image, consecutive windows appear to be the same [14].

We are interested in constrained texture synthesis or hole filling, in which the synthesized region must look like the surrounding texture, and the boundary between the new and old regions must be invisible.

In the problem, there are two major issues which need to be addressed: modeling of a texture, and sampling of texture from a given probability distribution.

3.2. Markov Random Field methods

Most of the texture synthesis methods make use of Markov Random Fields to come up with a distribution from where to pick the intensity value of a pixel.

Markov Random Field methods model a texture as a realization of a local and stationary random process. That is, each pixel of a texture image is characterized by a small set of spatially neighboring pixels, and this characterization is the same for all pixels.

Markov random fields define a powerful framework for specifying nonlinear interactions between features of the same nature or of a different one. They help to combine and organize spatial and temporal information by introducing strong generic knowledge about the features to be estimated.

They have been proven to cover a large variety of known textures and hence are the most widely used in synthesis.

Since there is no distinct classification for texture synthesis methods, unlike the image inpainting methods, I am presenting a summary of some of the widely referenced papers in this field.

3.3.1. Efros and Leung [15]

The pixels are grown from an initial seed, and they are grown pixel by pixel. For evaluating a synthesis metric, a neighbourhood around the unknown pixel value is taken, the size of which is chosen to correspond to the scale of the biggest regular feature expected in image.

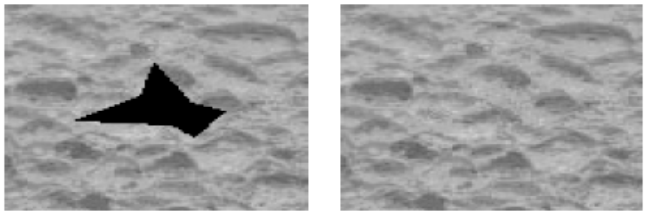

Fig 4: Hole-filling results from Efros-Leung [15]

The Efros-Leung algorithm suffers from the drawback that it is slow, can handle only frontal-parallel textures, and sometimes gives garbage growth in some image sectors. It is also a slow algorithm.

3.3.2. Wei-Levoy [14]

The goals of this implementation are to be efficient, general, user friendly and able to produce high quality, tileable textures.

It uses multiresolution pyramids, like Gaussian pyramids, which are hierarchy of low-pass filtered versions of the original image, such that successive levels correspond to lower frequencies. This use of multiresolution has to do with the fact that large scale structures need large pixel neighbourhoods to achieve good results in synthesis.

They also use a technique known as tree structured vector quantisation (TSVQ) to speed up the process. TSVQ causes blurring of edges, which is an unwanted effect. Performs relatively poorly on textures consisting of an arrangement of distinct objects of irregular but familiar shapes and sizes.

3.3.3. Ashikhmin [10]

Modification of the WL algorithm which performs better on this specific type of textures. It is significantly faster than the basic or accelerated WL algorithm even though coding complexity is about the same as the basic WL algorithm. The patches created by this algorithm have irregular shapes, which is useful for a variety of natural textures.

Another prominent approach is Harrison's resynthesizer [13].

4. Partial automation of restoration process

Multiresolution methods have been dealt with in [2] and [16] but in different contexts. They are used to improve the performance of inpainting and texture synthesis methods. The overview of multiresolution is as follows.

To reduce computational burden, the search for an appropriate neighbourhood is begun at a very coarse resolution. When a match is found, the set of matched windows is subjected to a higher resolution scan and so down to the point where we are satisfied with the degree of correlation between the unknown pixel's neighbourhood and the matched sets, and can choose the pixel colour value.

Shih et al [9] have developed an automated image inpaint tool. They use the variance of neighbourhood pixels and a multiresolution strategy to inpaint. The user has control over the process as a reversible control, in the sense that the user gains control to be able to undo changes which the automatic tool has already implemented. It uses PSNR as a metric in the multiresolution calculations.

The need for such kind of automation stems from the requirement for inpainting to be more accessible to the everyday user. As basic requirements, such a tool should be as automated as possible. Currently both texture synthesis and inpainting approaches to hole filling center around the assumption that the area to be filled in is well defined by the user. This can be a tedious process for an image with multiple distortions.

Instead, the reverse of multiresolution strategy and selective segmentation can be used to identify cracks in an image. Global image statistics such as average intensity and variance are calculated first.

The algorithm begins with a window of sufficiently high resolution. Within this window, it calculates the average pixel value and the variance measured against the mean of the entire image. If a weighted sum of these two values is much different from a similar weighted sum for the entire image, then the region is marked as a probable hole region.

We know for sure that a hole or crack in an image will show up if a segmentation algorithm is run over it.

At this stage, it is not known if the window contains a hole edge or if it is inside the hole. The resolution is now lowered, so we have a bigger window to look at, and gradient in both directions is calculated in this expanded window. A high gradient value indicates the presence of an edge in this expanded window. A segmentation algorithm is run for this expanded window. If the gradient values do not show a value beyond some threshold, then it is an indicator of the window still being inside the hole or crack, and further decrease in resolution takes place till we have a window large enough to give us gradient changes.

Windows in which wall following has already been carried out are not considered in successive iterations of the multiresolution algorithm. This avoids repetition. Also, at the next free location, the window resolution is back to a high value.

At the end of the process, the wall following data is merged to get continuous regions which have been marked as probable holes in the image. The user is returned a set of regions which he or she can accept or reject for the automated inpainting procedure to follow.

At present, this method has not been tested and remains a hypothesis. However, with proper threshold values for the gradients and with a well considered equation for weighting pixel intensity variance and the average intensity value, we should be able to automate inpainting further without

incurring too much of a time lag between giving the image as input and completing the inpainting or texture synthesis.

5. Future work and applications

Texture synthesis algorithms give excellent results in replicating patterns occurring in images, but stick to available data from input textures. Image inpainting makes use of local neighbourhood information and can produce pixel values which are not seen in the rest of the image given the use of isophotes. A natural amalgamation would be to tie the two together, and use inpainting algorithms which collaborate with texture synthesis to give improved restoration results [18].

The texture synthesis application of extrapolating a given photograph [15] can be used to produce a merging tool for 360 degree photographs which will not require the photograph frames to be exactly continuous.

Inpainting can be used in conjunction with 3d reconstruction and shape analysis to build an integrated robot system for excavation. The role of inpainting could be to fill in texture into an artifact which has been identified by the shape analysis, and 3d reconstruction will be given a complete and continuous textured object image to further process.

Acknowledgements

Dr. Birchfield, for teaching us various aspects of digital image processing, and briefly introducing us to texture synthesis after a lecture.

Vikram Iyengar and Brijesh Pillai, my original project group partners, for our discussion which led to the excavation application and ultimately to our individual projects.

References

1. **Digital inpainting based on the MumfordShah –Euler image model.** (2002)
S. Esedoglu and J. Shen. European J. Appl. Math., (2002)

2. **A Multiscale Method for Automated Inpainting** (2003)
R.J.Cant and C.S.Langensiepen. ESM2003

3. **Image Inpainting Implementation**
Sanjay G. Mavinkurve and Elijah M. Alper
http://www.eecs.harvard.edu/~sanjay/inpainting/

4. **Image Inpainting** (2000)
Marcelo Bertalmio et al. Siggraph 2000, Computer Graphics Proceedings (417-424)

5. **An Image Inpainting Technique based on the Fast Marching Method**.(2004)
A.Telea. Journal of Graphics Tools, vol. 9, no. 1, ACM Press, 2004

6. **Fast Digital Image Inpainting** (2001)
M. Oliveira, B. Bowen, R. McKenna, and Y. -S. Chang. In Proc. VIIP 2001, pp. 261—266, [CITY]: [PUB], 2001.

7. **Learning How to Inpaint from Global Image Statistics** (2003)
Anat Levin et al. Proceedings of the Ninth IEEE International Conference on Computer Vision (2003) - Volume 2 Page: 305

8. **Image selective smoothing and edge detection by nonlinear diffusion** (1992)
Catte et al. SIAM Journal on Numerical Analysis Volume 29 , Issue 1 (February 1992) Pages: 182 - 193

9. **An automatic image inpaint tool**. (2003)
Timothy K. Shih et al. ACM Multimedia 2003: 102-103

10. **Synthesizing Natural Textures** (2001)
Ashikhmin, M. 2001 Symposium on Interactive 3D Graphics, pages 217-226.

11. **Fast Texture Transfer** (2003)
Ashikhmin, M. IEEE Computer Graphics and Applications Volume 23 , Issue 4 (July 2003) Pages: 38 - 43

12. **Enhanced Texture Editing using Self Similarity**
S. Brooks, M. Cardle, and N.A. Dodgson. VVG2003

13. **A Non-Hierarchical Procedure for Re-Synthesis of Complex Textures** (2001)
Paul Harrison. {WSCG} 2001 Conference Proceedings

14. **Fast Texture Synthesis using Tree-structured Vector Quantization**. (2000)
Wei, L., Levoy, M. SIGGRAPH 2000 pp 479-488.

15. **Texture synthesis by non-parametric sampling**. (1999)
A. Efros and T. Leung. In International Conference on Computer Vision, volume 2, pages 1033–8, Sep 1999.

16. **Multiresolution sampling procedure for analysis and synthesis of texture images** (1997)
J. S. De Bonet.. In T. Whitted, editor, SIGGRAPH 97 Conference Proceedings, Annual Conference Series, pages 361–368. ACM SIGGRAPH

17. **Growing Fitted Textures** (2001)
Gabriele Gorla et al. SIGGRAPH 2001 Conference Abstracts and Applications, p. 191.

18. **GPU Image Inpainting via Texture Synthesis**
Hamilton Chong
www.people.fas.harvard.edu/~hchong/goodies/inpaint.pdf

19. **Non-Texture Inpainting by Curvature-Driven Diffusions (CCD)** (2000)
Chan, T., Shen, J. UCLA CAM TR 00-35, September 2000.