Ain shams University
Faculty of Computer &
Information Sciences
Computer Science
Department

Development of PDE-based Digital Inpainting Algorithm Applied to Missing Data in Digital Images

By
Yasmine Nader El-Glaly

B.Sc. in Computer & Information Sciences
(Computer Science)
2002

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master in Computer Science

Supervised by:

Prof. Essam Hamed Atta
Professor of Computer Sciences
Faculty of Computer &
Information Sciences
Ain Shams University

Prof. Taimor M. Nazmy
Professor of Computer Sciences
Faculty of Computer &
Information Sciences
Ain Shams University

Dr. Beih Sayed El Desouky
Assistant Professor of Mathematics
Faculty of Education
Suez Canal University

Cairo 2007

Abstract

Image inpainting is the process of filling in missing parts of damaged images based on information gathered from surrounding areas. In addition to problems of image restoration, inpainting can also be used in wireless transmission and image compression applications. In this thesis, we have developed an automatic digital inpainting system that enables the user to choose between two complementary approaches. The first is based on the solution of partial differential equation of isophote intensity to fill-in missing portions in the region under consideration, while the second is based on texture inpainting. The filling-in process is automatically done in regions containing completely different structures, textures, and surrounding backgrounds.

We have also presented an improved inpainting method based on the exemplar-based image inpainting technique. The developed method enhances the inpainting robustness and effectiveness by including image gradient and second derivative information during the inpainting process. Finally, we validated our developed method and compare the results with previous methods. Our results show that the developed algorithm can reproduce texture and at the same time keep the structure of the surrounding area of the inpainted region. The method proved to be effective in removing large objects from an image, ensuring accurate propagation of linear structures, and eliminating the drawback of “garbage growing” which is a common problem in other methods.

To my beloved parents and my dear husband

Acknowledgements

Thanks are due to ALLAH for getting this work done.

Many people contributed to produce this thesis in that form. I am supported, advised, encouraged and often inspired by my supervisors, family, friends and colleagues. I would like to thank them all. I guess that I should begin with my supervisors who did their best to guide me to develop and complete this honorific work. Many thanks to them and to each one individually; To Prof. Essam Atta, Prof. Taimor Nazmy, and Assistant Prof. Beih Desouky.

And finally, I thank the many, many people who patiently helped me, responded with creative and usable ideas and encouraged me to complete this work. I have nothing to say except “Thank You All”

List of Publications

Parts of this thesis are published as original papers in the conference proceedings, these are:

1. Y. El Glaly, E. Atta , T.M. Nazmy, B. Desouky, **Digital Inpainting for Cultural Heritage Preservation**, Proceedings of the 16th International Conference on Computer Theory and Application, ICCTA 2006, Alexandria, Egypt.

2. Y. El Glaly, E. Atta, T.M. Nazmy, B. Desouky, **Development of Combined Structure and Texture Inpainting Method**, Proceedings of the 3rd International Conference on Intelligent Computing and Information Systems, 2007, Cairo, Egypt.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

The story of inpainting begins in the art world. For centuries, people have been keenly interested in repairing missing sections of oil paintings, and doing so in a way that renders the restoration as imperceptible as possible (See Figure 1-1). However, differences of opinion regarding the best way to accomplish the retouching have been present from art restoration's inception.

The term inpainting is borrowed from paper art, where restoration artists are tasked with restoring faded and damaged paintings. In art however, the major concern is to hide the damage in whichever way complements the existing pigments and image the best, rather than repaint the damage parts of the painting since erasing paintings is generally not an option (that would be called overpainting) [1].

Image retouching ranges from the restoration of paintings to scratched photographs or films to the removal or replacement of arbitrary objects in images. Retouching can furthermore be used to create special effects (e.g., in movies). Ultimately, retouching should be carried out in such a way that when viewing the end-result it is impossible for an arbitrary observer, or at least very hard, to determine that the image has been manipulated or altered.

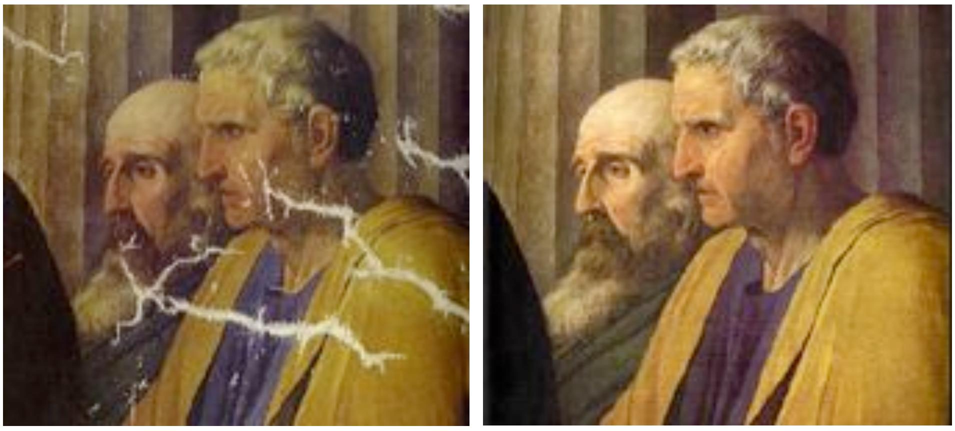


Figure 1-1: Example of manual inpainting performed by a professional artist.

Digital Inpainting is a term introduced in [2]. It alludes to how to perform inpainting digitally through image processing in some sense. Thereby also automating the process and reducing the interaction required by the user.

Ultimately, the only interaction required by the user is the selection of the region of the image to be inpainted.

Reference [2] describes the basic process of inpainting in four steps as follows:

1. The global picture determines how to fill in the gap, the purpose of inpainting being to restore the unity of the work.
2. The structure of the surroundings of the gap is continued into the gap, contour lines are drawn via the prolongation of those arriving at the boundary of the gap.
3. The different regions inside the gap, as defined by the contour lines are filled with color, matching those of the boundary of the gap.

4. The small details are painted (e.g., little white spots on an otherwise uniformly blue sky; in other words "texture" is added.

Partial differential equations (PDEs) are used for a large variety of image processing tasks, and recently, they have been proposed for so called inpainting techniques, which use PDE-based interpolation methods to fill in missing image data from a given inpainting mask.

Inpainting Vs. Denoising

Image inpainting is different than image denoising. Image inpainting is an iterative method for repairing damaged pictures or removing unnecessary elements from pictures. Classical image denoising algorithms don't apply to image inpainting. In common image enhancement applications, the pixels contain both information about real data and the noise, while in image inpainting, there is no significant information in the region to be inpainted. The information is mainly in the regions surrounding the areas to be inpainted. Another difference lies within the size of the data to be processed, the region of missing data in inpainting is usually large like long cracks in photographs, superimposed large fonts, and so on [3].

1.2 The Fundamentals of Digital Inpainting

Digital inpainting refers, as already mentioned, to inpainting through some sort of image processing. The digital inpainting process can be looked upon as a linear or non-linear transformation as illustrated in Figure 1-2, [4] and [5], where u_0 is the original image and u is the transformed image (i.e., the digitally inpainted image).

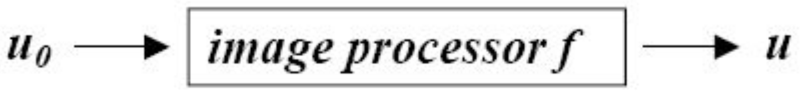


Figure 1-2 Linear transformation through an image processor f.

This is also how the concept of image processing is described in general [6]. The image processor can be looked upon as a function f as follows:

$$\text{f}: u_0 \rightarrow u$$

that is

$$u = \text{f}(u_0).$$

Now, let Ω denote the set of pixels (the region) of the image u_0 to be inpainted. Let ∂Ω denote the one pixel wide boundary of Ω so that (see also Figure1-3):

$$\Omega \subset u_0 \text{ , } \Omega = \{\text{the set of pixels of } u_0 \text{ to be inpainted}\}$$

and

$$\partial\Omega \subset u_0 \text{ , } \partial\Omega = \{\text{the boundary pixels of } \Omega\}$$

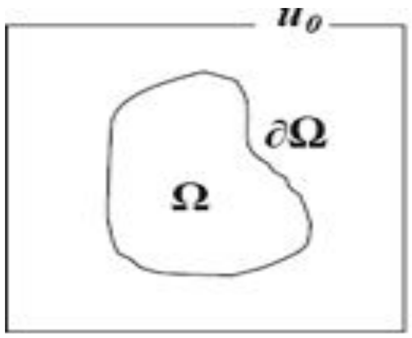


Figure 1-3 The image u_0 , the region Ω to be inpainted and its boundary ∂Ω.

The following steps describe the general solution to the problem (an explanation is given below):

```
STEP 1: SPECIFY $\Omega$
STEP 2: $\partial\Omega$ = THE BOUNDARY OF $\Omega$
STEP 3: INITIALIZE $\Omega$
STEP 4: FOR ALL PIXELS X, Y $\in \Omega$
            INPAINT X, Y IN $\Omega$ BASED ON INFORMATION IN $\partial\Omega$
```

The explanation is as follows: row 1 lets the user specify the region to be inpainted, row 2 computes the boundary of the region and row 3 initializes the region by for example, clearing existing color information. The for-loop "simply" inpaints the region based on information of its surroundings.

A first glance at the pseudo-code gives the impression that it is a piece of cake to implement. However, digital inpainting most often require a well thought-out strategy regarding the inpainting itself (i.e., the for-loop in this case). The general concepts of some of the existing approaches are described in chapter two.

A closely related area is the restoration of films, i.e., *image sequences* [7], [8], [9], [10] and [11]. The fundamental problem can be considered as being the same. However, the approach of digital inpainting regarding image sequences is significantly different from the approach of digital inpainting regarding still images. In order to inpaint Ω_n in frame n, where $n \in$ N, (i.e., a still image), information to inpaint Ω_n is derived out of adjacent frames, i.e., Ω_{n+k} from frame $n+k$, where $\{k: k \in$ Z, $k \neq 0\}$.

This, instead of using information found at the boundary $\partial\Omega_n$ of frame n. Hence, it is easy to realize that spatial and temporal changes such as the movement of an object must be taken into consideration when working with image sequences.

Digital inpainting regarding 3D-surfaces resembles digital inpainting of 2D images. Geometric partial differential equations (PDE's) are used to inpaint surface holes. However, instead of only working in two dimensions, the geometric PDE's may be used to inpaint surface holes in n dimensions [12] and [13].

1.3 Thesis Outline

In Chapter one, we gave a general overview about the digital inpainting problem, its definition, and its applications. The contents of the other chapters are outlined as follows:

Chapter Two

This chapter contains a survey of related work. The basic ideas and the concepts of some of the existing digital inpainting approaches are presented. The underlying theory is briefly explained.

Chapter Three

It contains our developed digital inpainting algorithm that is based on partial differential equations texture synthesis, which could successfully restore the texture as well as the structural data in the image. Also, we explain the details of digital inpainting algorithms that have been implemented. The motivation behind the choice of algorithms is also presented.

Chapter Four

It contains the results of our algorithm, and a comparison between our results and other inpainting algorithms results. Images representing typical inpainting cases are discussed, and the results are shown along with the computation times.

Chapter Five

In this chapter we present the thesis conclusions, and plans for future work.

Chapter 2

Related Work

The literature contains several inpainting algorithms that have been developed. They may roughly be divided into two categories:

1. Usually PDE based algorithms are designed to connect edges (discontinuities in boundary data) or to extend level lines in some adequate manner into the inpainting domain, see [14], [15], [16], [17], [18] and [19]. They are targeted on extrapolating geometric image features, especially edges. i.e. they create regions inside the inpainting domain. Most of them produce disturbing artifacts if the inpainting domain is surrounded by textured regions.

2. Texture synthesis algorithms use a sample of the available image data and aim to fill the inpainting domain such that the color relationship statistic between neighbored pixels matches those of the sample, see [20], [21], [22], [23], [24], [25], [26], [27] and [28]. They aim for creating intra–region details. If the inpainting domain is surrounded by differently textured regions, these algorithms can produce disturbing artifacts.

In this chapter, we will briefly explain the main ideas and the concepts of some of the existing inpainting algorithms.

2.1 PDE-based Inpainting Algorithm

Bertalmio et al. pioneered a digital image inpainting algorithm based on partial differential equations (PDEs) [2]. A user-provided mask specifies the portions of the input image to be retouched and the algorithm treats the input image as three separate channels (R, G and B). For each channel, it fills in the areas to be inpainted by propagating information from the outside of the masked region along level lines (*isophotes*). Isophote directions are obtained by computing at each pixel along the inpainting contour a discretized gradient vector (it gives the direction of largest spatial change) and by rotating the resulting vector by 90 degrees. This intends to propagate information while preserving edges.

A 2-D Laplacian is used to locally estimate the variation in color smoothness and such variation is propagated along the isophote direction. After every few steps of the inpainting process, the algorithm runs a few diffusion iterations to smooth the inpainted region. Anisotropic diffusion is used in order to preserve boundaries across the inpainted region.

Steady state is achieved if the smoothness of the image (its second derivative) is constant along the isophotes. The assumption of constant smoothness along isophotes is in general not justified. Since edges are continued straightly into the inpainting domain, round objects tend to develop straight segments meeting at acute angles, thus producing kinks and neglecting the principle of continuation of direction.

2.2 Texture-based Inpainting

PDE-based inpainting techniques work at the pixel level, and have worked well for small gaps, thin structures, and text overlays. However, for larger missing regions or textured regions, they may generate blurring artifacts.

The main point of interest in PDE based inpainting is a reasonable course of edges (discontinuities) which essentially form a one dimensional subset of the image. PDE inpainting algorithms usually fail if they are applied to a textured area or in areas containing regular patterns.

The reasons for failing are primarily the following:

1. Textures usually contain locally high gradients which may be misinterpreted as edges and thus are falsely continued into the inpainting domain.
2. The only image information that is used by PDE based inpainting is the boundary condition contained within a narrow band around the inpainting domain. Thus it is not at all possible to recognize regular patterns or structures from such a small amount of information.

Texture synthesis algorithms operate essentially on one pixel at a time and determine its value by looking for similar areas in the available image data. The fragment based algorithms can in some sense be considered as generalized texture synthesis. Instead of copying single pixels whole blocks are transferred into the inpainting domain thereby regarding that the resulting inpainting connects smoothly and is similar to the available image. Some hybrid algorithms which combine one or more techniques can also be used in inpainting domain [29].

In the following we give an overview on texture synthesis algorithms which have been used particularly for inpainting purposes:

Cant & Langensiepen [30] create copies of the image to be inpainted at various scales. In the coarsest image several candidates for a best matching patch are searched. In this search process also mirrored and rotated versions of the patches are considered. Once a set of candidate patch positions is found they are transferred to higher levels where the positions are adjusted to the finer resolution by searching in a neighborhood around the best positions found so far.

Thus an exhaustive search has only to be performed at the coarsest level, on the finer levels only a small subset of the image has to be considered.

Criminisi et.al. [31] and [32] use a confidence and a priority function. The priority of a pixel depends on the confidence and on the gradient magnitude of its surrounding. Pixels lying close to convex corners inside the inpainting domain get high priority since they are surrounded by many high confidence pixels and thus can be reliably inpainted. On the other hand pixels lying close to edges (high gradients) are also assigned high priority such that edges are treated preferably. Continuation of edges tends to build concave spikes into the inpainting domain and the priority of the surrounding pixels decreases. Thus a balanced growing of edges and texture patches is guaranteed. Patches are taken to be fixed size and constant shape (i.e., no rotation or mirroring is considered) and the similarity of patches is simply calculated using sum of squared differences.

2.3 Variational Image Inpainting

A different approach to inpainting is proposed by Chan and Shen [33]. It is a variational-based method. An Euler-Lagrange equation is used and the inpainting of Ω is performed by using anisotropic diffusion. It is targeted at handling images that do not contain intense texture structures (e.g., natural images). The authors emphasize that any possible solution to the inpainting problem is only a good approximation or a “best” guess, i.e., it is more or less impossible to completely restore every detail of Ω. The “best” guess is modeled by the optimization of some energy or cost functional. The interpolation is limited to creating straight isophotes.

Furthermore, the authors distinguish the inpainting problem into two levels, the local and global level. The method only relies on local information for the inpainting of Ω. This approach also takes into account the sampling theorem [34]. The analogy with inpainting is that in order to get an accurate reconstruction the sampling distance has to be small enough, i.e., Ω has to be small. Thus, the method is developed for small Ω. Another reason for focusing on small Ω is the fact that it is somewhat difficult and thereby also computationally expensive to catch global patterns due to its intense variations in both scale and structure [33].

The principle behind their approach can be summarized as follows: Variational denoising and segmentation models all have an underlying notion of what constitutes an image. In the inpainting region, the models of Chan and Shen reconstruct the missing image features by relying on these built-in notions.

As an extension of [33], Chan and Shen, describe a Curvature Driven Diffusion (CDD) approach in [35]. It extends the previously described method and like its predecessor it is a PDE-based method. The extension is aimed at handling larger Ω. It does this by taking into account geometric information of the isophotes when defining the “strength” of the diffusion process [36]. The diffusion gets stronger where the isophotes are having a larger curvature, while it dies away as the isophotes stretch out.

The CDD approach may be looked upon as being orthogonal to the first inpainting method described in section 2.1. While the latter one propagates smoothness along the isophotes, this approach diffuses pixel information along the normal direction (i.e., perpendicular to the isophotes).

This first model introduced by Chan and Shen used the total variation based image denoising model of Rudin, Osher, and Fatemi [37] for the inpainting purpose. The model can successfully propagate sharp edges into the damaged domain. However, because the regularization term in this model exacts a penalty on the length of edges, this technique cannot connect contours across very large distances. Another caveat to the method is that it does not always keep the direction of isophotes continuous across the boundary of the inpainting domain.

Subsequently, Kang, Chan, and Shen [38] introduced a new variational image inpainting model that addressed the shortcomings of the total variation based one. The model is motivated by the work of Nitzberg, Mumford, and Shiota [39], and includes a new regularization term that penalizes not merely the length of edges in an image, but the integral of the square of curvature along the edge contours.

This allows both for isophotes to be connected across large distances, and their directions to be kept continuous across the edge of the inpainting region.

2.4 Simultaneous Structure and Texture Image Inpainting

Various algorithms have been proposed to fill missing regions with available information from their surroundings. In cases of texture synthesis the information required for texture generation is from the input image. Since most image areas are not just pure texture or pure structure, this approach provides just a first attempt in the direction of simultaneous texture and structure filling-in.

The basic idea of the algorithm of Bertalmio et al. [40] is that first decomposing the original image into the sum of two images, one capturing the basic image structure and the other capturing the texture

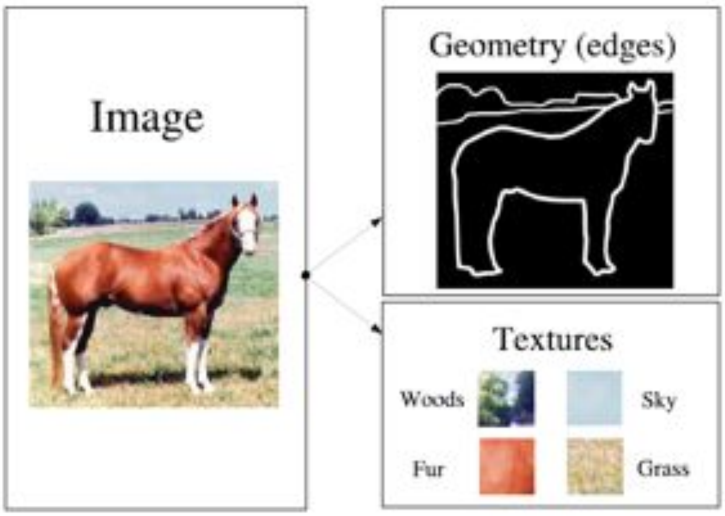


Figure 2-1 Decomposition of an image into geometry and texture

(and random noise inside). The first image (structure image) is inpainted following the work by Bertalmio et al. [1], while the other one is filled-in with a texture synthesis algorithm following the work by Efros et al. [41].

The two reconstructed images are then added back together to obtain the final reconstruction of the original data. In other words, the general idea was to perform both structure inpainting and texture synthesis not on the original image, but on a set of images with very different characteristics that are obtained from decomposing the given image. The decomposition produced images suited for these two reconstruction algorithms. The algorithm works well enough for well designed structures in the image, but in case of natural images the structures do not have well defined edges so the results might not be correct. Also for large unknown regions the algorithm might not give plausible results.

2.5 Inpainting Using Navier-Stokes Equations

In the method described in [3] Bertalmio et al. modified their method through an analogy of the Navier-Stokes and a slightly different underlying mathematical model. The Navier-Stokes equations are non-linear PDE's. Employing these equations it is possible to describe fluid dynamics, e.g., ocean currents, water flow, movement of air in the atmosphere and other phenomena.

The method is directly based on the Navier-Stokes equations for fluid dynamics, which has the immediate advantage of well-developed theoretical and numerical results. This is a new approach for introducing ideas from computational fluid dynamics into problems in computer vision and image analysis.

This approach [3] uses ideas from classical fluid dynamics to propagate isophote lines continuously from the exterior into the region to be inpainted. The main idea is to think of the image intensity as a stream function for a two-dimensional incompressible flow. The Laplacian of the image intensity plays the role of the vorticity of the fluid; it is transported into the region to be inpainted by a vector field defined by the stream function. The resulting algorithm is designed to continue isophotes while matching gradient vectors at the boundary of the inpainting region.

Both the inviscid and viscous problems, with appropriate boundary conditions, are globally well-posed in two space dimensions. Solutions exist for any smooth initial condition and they depend continuously on the initial and boundary data.
In terms of the stream function, the Laplacian of the stream function, and hence the vorticity, must have the same level curves as the stream function. The analogy to image inpainting is now clear: the stream function for inviscid fluids in 2D satisfies the same equation as the steady state image intensity equation.

The point is that, in order to solve the inpainting problem, we have to find a steady state stream function for the inviscid fluid equations, which is a problem possessing a rich and well developed history.
The main analogy that this approach is built on is the parallelism between the stream function in a 2D incompressible fluid and the role of image intensity function "I" in the inpainting algorithm. This allows us to design a new inpainting method that will achieve the same steady equation.

Let Ω be a region in the plane in which we want to inpaint from surrounding data. Assume that the image intensity I_0 is a smooth function (with possibly large gradients) outside of Ω and we know both I_0 and ΔI_0 on the boundary $\partial\Omega$.

The authors design a 'Navier-Stokes' based method for image inpainting. In this method the fluid dynamic quantities have the following parallel to quantities in the inpainting method.

Navier Stokes	Image inpainting
Stream function ψ	Image intensity I
Fluid velocity V= $\nabla^{\perp}\psi$	Isophote direction $\nabla^{\perp}$I
Vorticity $\omega = \Delta\psi$	Smoothness $\omega = \Delta$ I
Fluid viscosity ν	Anisotropic diffusion ν

The goal is to solve a form of the Navier-Stokes equations in the region to be inpainted. In fluid problems with small viscosity, the above dynamics can take a long time to converge to steady state, making the method less practical. Instead there are pseudo-steady methods that involve replacing the Poisson equation with a dynamic relaxation equation.

The existence of viscosity in the equations produces diffusion which can result in a blurring of sharp interfaces, and image gradients in the inpainting region. Hence it is often desirable to include anisotropic diffusion either added directly to the dynamical problem or as an additional step in conjunction with the Poisson step.

This analogy also shows why diffusion is required in the original inpainting problem. The natural boundary conditions for inpainting are to match the image intensity on the boundary of the inpainting region and also the direction of the isophote lines which for the fluid problem is effectively a generalized boundary condition that requires a Navier-Stokes formulation, introducing a diffusion term. In practice nonlinear diffusion (as in Perona-Malik [36], and Rudin, Osher, Fatemi [37]) works very well to avoid blurring of edges in the inpainting.

2.6 Inpainting Using the Vector Valued Ginzburg-Landau Equation

Another inpainting approach is based on the complex Ginzburg–Landau equation [42]. The use of this equation is motivated by some of its remarkable analytical properties. While common inpainting technology is especially designed for restorations of two dimensional image data, the Ginzburg–Landau equation can straight forwardly be applied to restore higher dimensional data, which has applications in frame interpolation, improving sparsely sampled volumetric data and to fill in fragmentary surfaces. The latter application is of importance in architectural heritage preservation.

The Ginzburg–Landau equation is originally developed by Ginzburg and Landau [43] to phenomenologically describe phase transitions in superconductors near their critical temperature. The equation has proven to be useful in several distinct areas besides superconduction.

Solutions of the real valued Ginzburg–Landau equation develop homogeneous areas, which are separated by phase transition regions that are interfaces of minimal area. In image processing homogeneous areas correspond to domains of constant grey value intensities, and phase transitions to edges. Thus the quoted properties make the real valued Ginzburg–Landau equation a reasonable method for high quality inpainting of binary images.

The authors calculate a complex valued function $\tilde{g}$, whose real part is the geometry image g scaled to a range of values between -1 and 1 and the imaginary part is nonnegative.

The solution of the Ginzburg–Landau equation reveals high contrast in the inpainting domain, which makes it particularly suited for inpainting purposes. However, the level lines of the solution of the Ginzburg–Landau equation at the boundary of the inpainting domain might look kinky. The kinks can be smoothed via coherence enhancing diffusion.

The Ginzburg–Landau equation has some favorable properties for image processing applications. For instance it has a tendency to connect discontinuities by minimal surface interfaces. So given an image with a missing or unwanted domain Ω the algorithm described in [42] finds a solution of the Ginzburg–Landau equation, where the available parts of the image are prescribed as Dirichlet boundary condition.

The numerical experiments have shown that best results are obtained for locally small inpainting areas which make this approach well suited for removing cracks or superimposed texts.

The Ginzburg-Landau algorithm does usually not give good results when the inpainting domain is large or surrounded by strongly textured regions. This is a drawback commonly shared by all PDE based inpainting algorithms. Instead they perform well if the inpainting domain is locally small, e.g., consists of small stains or thin structures.

Chapter 3

Enhanced PDE Digital Inpainting Algorithm

Image inpainting is an iterative method for repairing damaged pictures or removing unnecessary elements from pictures. This activity consists of filling in the missing areas or modifying the damaged images in a non-detectable way by an observer not familiar with the original images. The development of our PDE-based inpainting method follows the work of Sapiro et al. [2].
The following sections describe the method details and its implementation

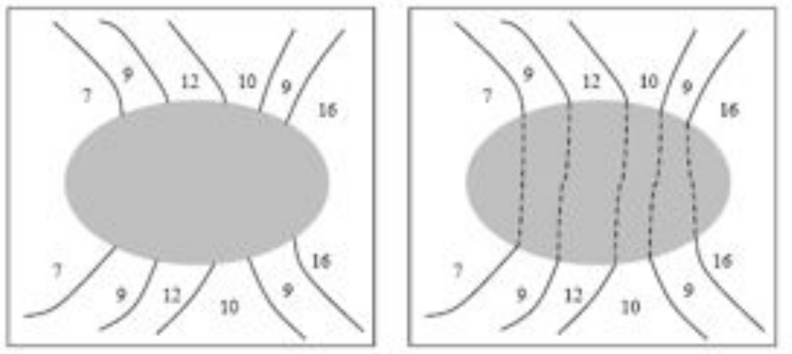
3.1 PDE-based Digital Inpainting Algorithm

The image inpainting technique is based on filling holes in images by propagating linear structures (called isophotes in the inpainting literature) into the target region via diffusion (figure 3-1). This is analogous to the partial differential equations (PDE) of physical heat flow. Color images are considered as a set of three images, and for each one, the region is filled by propagating information from the outside of the masked region along level contours (isophotes).

Isophote direction is obtained by computing the discretized gradient vector of each pixel along the contour (the gradient indicates the direction of largest color change) and by rotating the resulting vector by 90 degrees.
The isophote direction is defined as

$$\nabla^{\perp} I \ (i,j) = (\frac{\partial}{\partial j}, \frac{\partial}{\partial i}) I(i,j)$$

where I (i, j) is the image data (gray scale value) at a point (i, j) in the two dimensional picture. Image data from the boundary of the inpainting region is then propagated a short distance into the inpainting region along these isophotes.



(a) **(b)**

Figure 3-1: The selected region represented by the grey ellipse. (The numbers represent the grey levels)

(a) Incomplete level lines.

(b) Inpainted level lines with curved connections.

This intends to propagate information while preserving structures. A 2-D Laplacian operator is used to locally estimate the variation in smoothness. Through propagating such variation, the isophote direction is obtained.

The PDE to solve is then

$$\frac{\partial I}{\partial t} + \nabla^{\perp} I . \nabla \Delta I = 0$$

This has the effect of propagating the smoothness operator ΔI into the inpainting region.

After every step of the inpainting process, a few diffusion iterations are taken to smooth the inpainted region. Anisotropic diffusion is used in order to preserve edges across the inpainted region [36].

Every few steps in the numerical process, an iteration of anisotropic diffusion is added by calculating

$$\frac{\partial I}{\partial t} = k(i,j,t)\left|\nabla I(i,j,t)\right|$$

at all points within the inpainting region. Here, k (i, j, t) is the Euclidean curvature of the two dimensional surface I (i, j, t) at the point (i, j). This diffusion process allows the successive isophote lines to curve, if need be, as they are propagated.

3.1.1 Numerical Implementation of the Inpainting Algorithm

The inpainting algorithm goes as follows,

Input:

- Image to be inpainted,
- Mask that delimits the position to be inpainted.

Pre-processing step:

Whole original image undergoes anisotropic diffusion smoothing. (To minimize the influence of noise on the estimation of the direction of the isophotes arriving at $\partial\Omega$)

Inpainting loop:

Repeat the following until a steady state is achieved. (Only values inside Ω are modified)

Every few iterations, a step of anisotropic diffusion is applied.

$$I^{n+1}(i,j) = I^{n}(i,j) + \Delta t\, I_t^{n}(i,j), \forall (i,j) \in \Omega \qquad (1)$$

$$I_t^n(i,j) = \left(\overrightarrow{\delta L^n}(i,j) \cdot \frac{\overrightarrow{N}(i,j,n)}{\left|\overrightarrow{N}(i,j,n)\right|} \right) \left|\nabla I^n(i,j)\right| \tag{2}$$

$$\overrightarrow{\delta L^n}(i,j) = (L^n(i+1,j) - L^n(i-1,j), L^n(i,j+1) - L^n(i,j-1)) \tag{3}$$

$$L^n(i,j) = I_{xx}^n(i,j) + I_{yy}^n(i,j) \tag{4}$$

$$\frac{\overrightarrow{N}(i,j,n)}{\left|\overrightarrow{N}(i,j,n)\right|} = \frac{\left(-I_y^n(i,j), I_x^n(i,j)\right)}{\sqrt{\left(I_x^n(i,j)\right)^2 + \left(I_y^n(i,j)\right)^2}} \tag{5}$$

$$\beta^n(i,j) = \overrightarrow{\delta L^n}(i,j) \cdot \frac{\overrightarrow{N}(i,j,n)}{\left|\overrightarrow{N}(i,j,n)\right|} \tag{6}$$

$$\left|\nabla I^n(i,j)\right| = \begin{cases} \sqrt{(I_{xbm}^n)^2 + (I_{xfM}^n)^2 + (I_{ybm}^n)^2 + (I_{yfM}^n)^2} \, where \, \beta^n > 0 \\ \sqrt{(I_{xbM}^n)^2 + (I_{xfm}^n)^2 + (I_{ybM}^n)^2 + (I_{yfm}^n)^2} \, where \, \beta^n < 0 \\ 0 \, where \, \beta^n = 0 \end{cases} \tag{7}$$

β^n (Eq.6) is the projection of (Eq.3) onto the normalized normal vector (Eq.5) (the isophote direction), where (Eq.4) is the smoothness estimation (the Laplacian). (Eq.7) is a slope-limited version of the norm of the gradient of the image. The subscript indexes b and f denote the backward and forward differences, while m and M denote the minimum and maximum between the derivative and zero.

I_{xf} = I (i +1, j) – I (i, j) Forward difference in the x-direction

I_{xb} = I (i, j) – I (i -1, j) Backward difference in the x-direction

I_{yf} = I (i, j+1) – I (i, j) Forward difference in the y-direction

I_{yb} = I (i, j) – I (i, j-1) Backward difference in the y-direction

Δ t (Eq.1) may be looked upon as a speed factor (i.e., small Δ t gives makes the algorithm converge slower). The algorithm runs in a total of T iterations. The inpainting itself (Eq.1) runs in a total of A iterations steps, and B diffusion iterations are performed after the A inpaintings.

3.1.2 Anisotropic Diffusion Algorithm

The implementation of discrete anisotropic diffusion is based on the formulas shown in Perona and Malik's literature, Scale-Space and Edge Detection Using Anisotropic Diffusion [36].

$$I^{n+1}(i,j)=I^{n}(i,j)+\lambda\left[C_N\nabla_N I(i,j)+C_S\nabla_S I(i,j)+C_E\nabla_E I(i,j)+C_W\nabla_W I(i,j)\right]$$

The subscripts N, S, E, and W are corresponding to direction north, south, east and west respectively. Where

$$\nabla_N I^n(i,j) = I^n(i-1,j) - I^n(i,j)$$

$$\nabla_S I^n(i,j) = I^n(i+1,j) - I^n(i,j)$$

$$\nabla_E I^n(i,j) = I^n(i,j+1) - I^n(i,j)$$

$$\nabla_W I^n(i,j) = I^n(i,j-1) - I^n(i,j)$$

$$C_N{}^n(i,j) = \mathrm{g}\,(|\nabla_N I^n(i,j)|)$$

$$C_S{}^n(i,j) = g(|\nabla_S I^n(i,j)|)$$

$$C_E{}^n(i,j) = g(|\nabla_E I^n(i,j)|)$$

$$C_W{}^n(i,j) = g(|\nabla_W I^n(i,j)|)$$

$$g(\nabla I) = e^{(-\|\nabla I\|/K)^2}$$

The value of λ is fixed at 0.25 and K is selected between 0.3 and 0.5.

Although this algorithm succeeds to inpaint small regions (figure 3-2) in non detectable way, it can not reproduce texture as well. When we run this algorithm to inpaint relatively large regions, it introduces blurring in the image (figure 3-3).

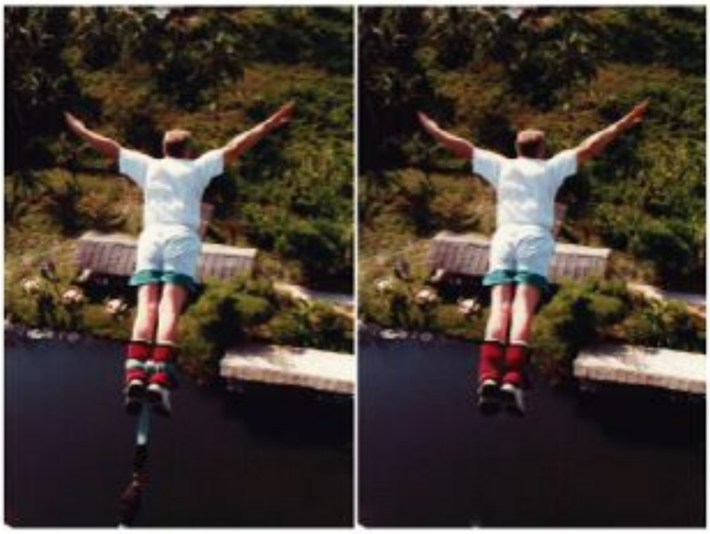


Figure 3-2: The bungee cord and the knot tying the man's feet have been removed.

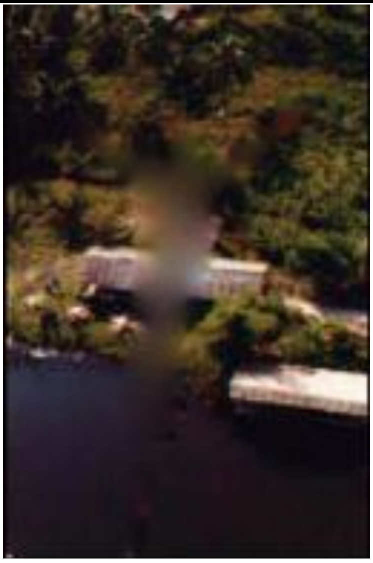


Figure 3-3: Limitations of the algorithm: texture is not reproduced.

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 (figure 3-4).

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. We modified the texture synthesis approach used by Criminisi et al.[31] to accomplish this task.

In Criminisi et al 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. The algorithm uses a best-first algorithm in which the confidence in the synthesized pixel values is propagated in a manner similar to the propagation of information in PDE inpainting algorithm.

Criminisi et al. use the sampling concept from Efros' approach [41]. 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.

This algorithm, however, has some problems: firstly, it merely adopts a simple priority computing strategy without considering the accumulative matching errors; secondly, the matching algorithm for texture synthesis only uses the color information; thirdly, the filling scheme just depends on the priority disregarding the similarity. As a result of lacking robustness, their algorithm sometimes runs into difficulties and “grows garbage”.

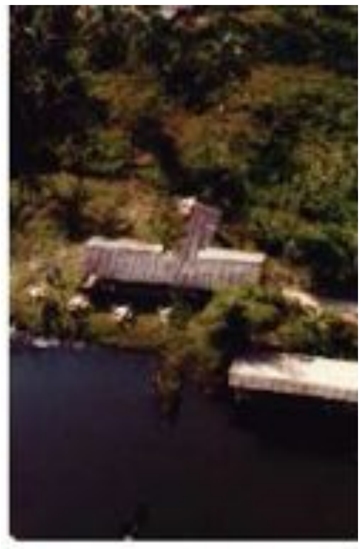


Figure 3-4: The linear structure is not well preserved

To solve these problems, we propose an enhanced exemplar-based inpainting algorithm.

3.2 Enhanced Exemplar-based Inpainting Algorithm

The exemplar-based inpainting algorithm consists mainly of three iterative steps, until all pixels in the inpainted region are filled. The region to be filled, i.e., the target region is indicated by Ω, and its contour is denoted $\partial\Omega$. The contour evolves inward as the algorithm progresses, and so we also refer to it as the “fill front.” The source region which remains fixed throughout the algorithm, provides samples used in the filling process (figure 3-5). In order to find the most similar patch in the source region to the target patch, we search the whole source image to find the best fit. The similarity is measured by computing the sum of squared distance in color between each corresponding pixel in the two patches.

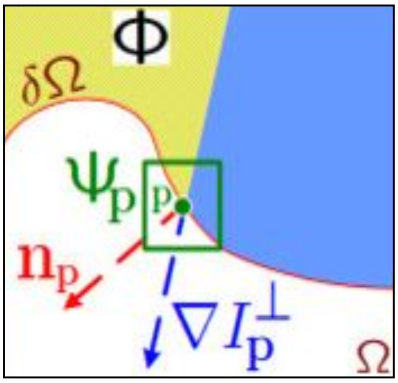


Figure 3-5: Notation diagram. Given the patch Ψ_p, n_p is the normal to the contour $\partial\Omega$ of the target region Ω and $\nabla I_p^{\perp}$ is the isophote (direction and intensity) at point p. The entire image is denoted with I.

3.3 Modifying the Distance Function

The similarity measure based only on color is insufficient to propagate accurate linear structures into the target region, and leads to garbage growing. So, we add to this distance function a new term G representing image gradient as an additional similarity metric.

$$G = G(\Psi_p) - G(\Psi_q)$$

Where G is the gradient value for each pixel in the two considering patches. Hence, the similarity function now depends on the difference between the patches according to two criteria, the difference in color and in gradient values.

The gradient of an image measures how it is changing. It provides two pieces of information. The magnitude of the gradient tells us how quickly the image is changing, while the direction of the gradient tells us the direction in which the image is changing most rapidly.

The details of the algorithm implementation is as follows,

1. Computing patch priorities

Given a patch Ψ_p centered at the point p for some $p \in \partial\Omega$, its priority P (p) is defined as the product of two terms:

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

C (p) is the confidence term and D (p) is the data term, and they are defined as follows:

$$\mathrm{C(p)} = \frac{\sum_{q \in \Psi_p \cap \bar{\Omega}} \mathrm{C(q)}}{|\Psi_p|} \quad \text{, and} \qquad D(p) = \frac{|\nabla I_p^{\perp} \cdot n_p|}{\alpha}$$

Where $|\Psi_p|$ is the area of Ψ_p, α is a normalization factor (e.g., $\alpha = 255$ for a typical grey-level image), and n_p is a unit vector orthogonal to the front $\partial\Omega$ in the point p. The priority is computed for every border patch, with distinct patches for each pixel on the boundary of the target region. The patch with the highest priority is the target to fill.

2. Propagating structure and texture information

Search the source region to find the patch which is most similar to $\Psi_{p^\wedge}$. Formally, $\Psi_{q^\wedge} = \arg\min_{\Psi_q \in \Omega} d(\Psi_{p^\wedge}, \Psi_q)$

The distance $d(\Psi_a, \Psi_b)$ between two generic patches $\Psi_a and\ \Psi_b$ is simply defined as the sum of squared differences (SSD) of the already filled pixels in the two patches.

$$d = \sum_{i=1}^{i=A} (I_{ai} - I_{bi})^2 + (G_{ai} - G_{bi})^2$$

Where, G presents the image gradient vector, I is the RGB colour vector, D is the distance (the larger d is, the less similar they are), and A is the known pixels number in $\Psi_{p^\wedge}$.

Having found the source exemplar $\Psi_{q^\wedge}$, the value of each pixel to be filled, is copied from its corresponding position.

3. Updating confidence values

The confidence C (p) is updated in the area delimited by $\Psi_{p^\wedge}$, as follows:

$$c(q) = c(p^\wedge)\ \forall q \in \Psi_{p^\wedge} \cap \Omega$$

As filling proceeds, confidence values decay, indicating that we are less sure of the color values of pixels near the centre of the target region.

3.4 Modifying the Data Term

As observed in the previous section, the results still need more enhancements. The problem in previous algorithms is the way of computing the patch priorities. In exemplar-based approaches, filling order is critical because the quality of inpainting result is highly influenced by the order of the filling process.

In Figure 3-6, the ideal order of filling is shown. In traditional method, filling order is evaluated by the priority, which only considering the confidence value of a pixel. The information which the confidence value of a pixel could give us is not enough, since we can't know the patch's "*real, visual*" clues such as *isophotes* or how regular the patch's color distribution is.

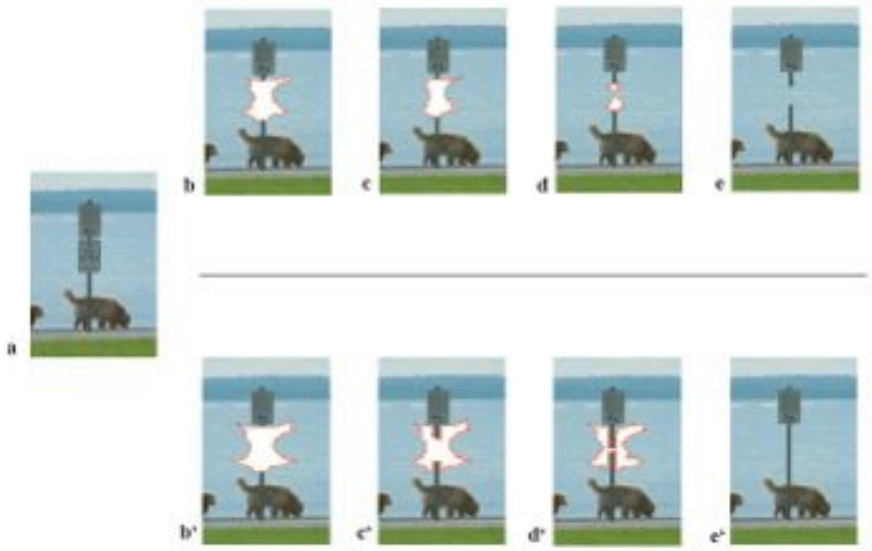


Figure 3-6: The comparison between concentric layer filling and desired filling order behavior [32]. ***(a) The original image. (b, c, d ,e) The completion steps in concentric filling order. (b', c', d', e') The completion steps in desired filling order. In (b, c, d, e), as lack of attention on the line structures, the result may not be desirable.***

The patch priority is defined mainly by the confidence term, where the confidence term is computed by the number of original pixels in the source patch. However, that leads to the “onion peel” method, where the target region is synthesized from the outside inward, in concentric layers. That gives in the end unsatisfying results that totally lose the linear structure of the image.

To solve this filling order problem, Criminisi et al. suggests a new term to determine the priority of the patch, in combination with the confidence term which is the data term. The data term according to them is the isophote direction $\nabla I_p^{\perp}$ multiplied by a unit vector orthogonal to the fill front $\partial\Omega$ at the point p. This new definition of the priority patch enhances the results because it forces the algorithm to take into consideration the isophote direction while propagating the information from the source region into the target region.

We have tested this algorithm, and it gives satisfying results in the texture-based images that have simple linear structures. But if the images to be inpainted are more complicated, containing sophisticated structures of similar textures, the algorithm fails.

After extensive testing, there have been a few cases where the exemplar-based algorithm in some subjective sense fails. Two of these cases are shown in Figure 3-7. It is clear that the algorithm sometimes make bad choices of choosing the substitute patch. As Figure 3-7(c) shows, the algorithm has chosen patches from the mountain. This depends on the assigned priority values (and thus the filling order). For example, the pixels in the region near the mountain have been given a higher priority and have been filled first. This has continued throughout the iterations and therefore the best-matching mountain patches have progressively expanded patches from the mountain up in the air. Figure 3-7 (e) experiences the same dilemma.

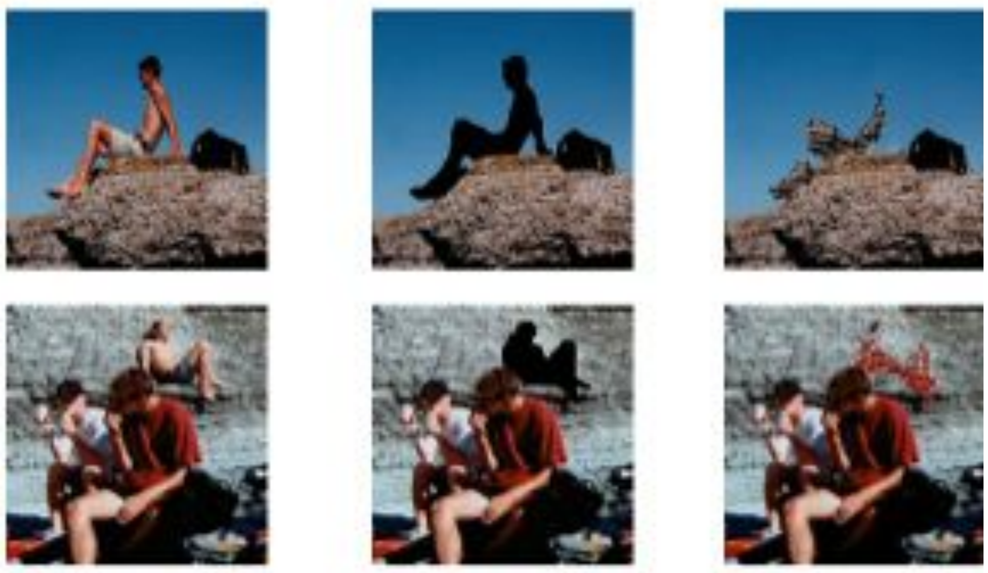


Figure 3-7 Two examples of algorithm failure [32]

(a-c) The original, The mask and the result
(d-f) The original, The mask and the result

We address this problem in this section, trying to develop an algorithm that can deal with such difficult cases.

3.4.1 SDGD Overview

By reviewing the literature, we find that the second derivative in the gradient direction (SDGD) has properties that can be very useful in our inpainting problem. The SDGD (also called the second directional derivative), a nonlinear operator, can be expressed in first and second derivatives. Haralick [44] approximated the SDGD using derivatives of a cubic polynomial model approximation of the underlying grey level surface. The cubic fit is equivalent to different low-pass filters for different derivatives, so that Haralick's result cannot be interpreted as the SDGD of some linear shift-invariant low-pass filtered image. Clark [45] and Torre [46] used the SDGD in its analytical description. Very fast approximations of the SDGD can be obtained using local maximum and minimum filters (grey-scale dilation and grey-scale erosion) ([47]; [48]; [49]; [50]). A quantitative evaluation between an analytical SDGD, a non analytical SDGD and the Marr-Hildreth operator [50] shows comparable performance on synthetic test images heavily disturbed by Gaussian noise.

SDGD filter - A filter that is especially useful in edge finding and object measurement is the Second-Derivative-in-the-Gradient-Direction (SDGD) filter. This filter uses five partial derivatives:

$$I_{xx} = \frac{\partial^2 I}{\partial x^2} \qquad I_{xy} = \frac{\partial^2 I}{\partial x \, \partial y} \qquad I_x = \frac{\partial I}{\partial x}$$

$$I_{yx} = \frac{\partial^2 I}{\partial x \, \partial y} \qquad I_{yy} = \frac{\partial^2 I}{\partial y^2} \qquad I_y = \frac{\partial I}{\partial y}$$

Note that $I_{xy} = I_{yx}$ which accounts for the five derivatives.

This SDGD combines the different partial derivatives as follows:

$$\text{SDGD} = \frac{I_{xx} I_x^2 + 2 I_{xy} I_x I_y + I_{yy} I_y^2}{I_x^2 + I_y^2}$$

Where I_{nn} = B * G_{nn},

nn = x, y, xx, yy or xy

and * represents convolution in the spatial domain. Here B is the image array whose Fourier transform has been band-limited by multiplying it by a low-pass Gaussian filter.

G_x and G_y are the first derivatives, with respect to x and y respectively, of a Gaussian

$$G = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{r^2}{2\sigma^2})$$

of width σ with pixel radial location r, given by $r^2 = x^2 + y^2$ where x and y are the pixel coordinates relative to the centre of the array. G_{xx}, G_{yy} and G_{xy} are the second derivatives.

As one might expect, the large number of derivatives involved in this filter implies that noise suppression is important and that Gaussian derivative filters, both first and second order are highly recommended if not required. The standard deviation sigma is the only parameter of the Gaussian filter; it is proportional to the size of neighborhood on which the filter operates. Pixels more distant from the center of the operator have smaller influence, and pixels further from the center have negligible influence.

From figure 3-8, we can see the effect of the SDGD of Gaussian after applying it to the image. It demonstrates how the SDGD can detect the curved edges precisely.

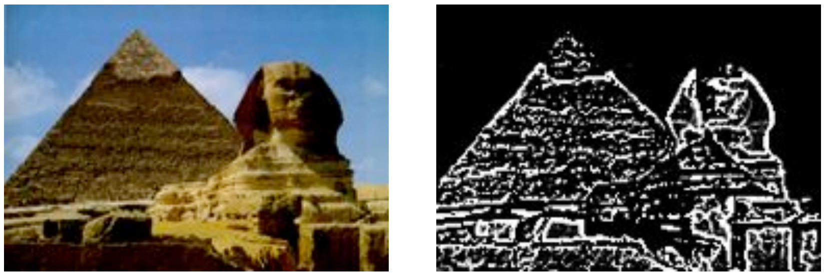


Figure 3-8 (a) Image of the pyramid. (b) Image after SDGD filter.

3.4.2 The Developed Algorithm

The second derivative in gradient direction (SDGD) produces a predictable bias in edge location towards the centers of curvature, so it can be used to determine the fill order.

The details of the algorithm implementation is as follows,

1. Computing patch priorities

Given a patch Ψ_p centered at the point p for some $p \in \partial\Omega$, its priority P (p) is defined as the product of two terms:

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

C (p) is the confidence term, it means how many information we can trust in the pixel *p*'s neighborhood. It can be written as

$$C(p) = \frac{\sum_{q \in \Psi_p \cap \bar{\Omega}} C(q)}{\left|\Psi_p\right|} \qquad (8)$$

Where $\left|\Psi_p\right|$ is the area of Ψ_p

And D (p) is the data term, it is defined as follows:

$$D(p) = \frac{I_{xx} I_x^2 + 2 I_{xy} I_x I_y + I_{yy} I_y^2}{I_x^2 + I_y^2} \qquad (9)$$

Where

$$I_{xx} = \frac{\partial^2 I}{\partial x^2} \quad I_{xy} = \frac{\partial^2 I}{\partial x\, \partial y} \quad I_x = \frac{\partial I}{\partial x}$$

$$I_{yx} = \frac{\partial^2 I}{\partial x\, \partial y} \quad I_{yy} = \frac{\partial^2 I}{\partial y^2} \quad I_y = \frac{\partial I}{\partial y}$$

$I_{nn} = \mathrm{B} * G_{nn}$,

nn = x, y, xx, yy or xy

and * represents convolution in the spatial domain, B is the image, G_x and G_y are the first derivatives, with respect to x and y respectively, of a Gaussian

$$G = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{r^2}{2\sigma^2})$$

of width σ with pixel radial location r, given by $r^2 = x^2 + y^2$ where x and y are the pixel coordinates relative to the centre of the array. G_{xx}, G_{yy} and G_{xy} are the second derivatives.

During initialization, the function C(p) is set to C(p) = 0 for all point p in the unknown area Ω, and C(p) = 1 for all point p in the known area *I* - Ω. The confidence term C (p) may be considered as a measure of the amount of reliable information surrounding the pixel p. The intention here is to fill first those patches which have more of their pixels already filled, with additional preference given to pixels that were filled early on (or that were never part of the target region).

At a coarse level, the term C (p) of Eq.8 approximately enforces the desirable concentric fill order. As filling proceeds, pixels in the outer layers of the target region will tend to be characterized by larger confidence values, and therefore be filled earlier; pixels in the center of the target region will have lesser confidence values.

The data term D (p) of Eq.9 is a function of the curvature of isophotes hitting the front $\partial\Omega$ at each iteration. This term enforces the priority of a patch to get a higher value when an isophote happens to "flow" into. This term has crucial importance in our algorithm because it pays large attention on linear structures to encourage linear structures to be synthesized first, and, therefore propagated securely into the target region. Broken curves tend to connect thus fitting in with the instincts of human beings.

The priority is computed for every border patch, with distinct patches for each pixel on the boundary of the target region. The patch with the highest priority is the target to fill.

2. Propagating structure and texture information

Once all priorities on the fill front have been computed, the patch $\Psi_{p^\wedge}$ with highest priority is found. We then fill it with data extracted from the source region Ω. we propagate image texture by direct sampling of the source region.

Search the source region to find the patch which is most similar to $\Psi_{p^\wedge}$. Formally,

$$\Psi_{q^\wedge} = \arg\min_{\Psi_q \in \Omega} d(\Psi_{p^\wedge}, \Psi_q)$$

The distance $d(\Psi_a, \Psi_b)$ between two generic patches Ψ_a *and* Ψ_b is simply defined as the sum of squared differences (SSD) of the already filled pixels in the two patches.

$$d = \sum_{i=1}^{i=A} (I_{ai} - I_{bi})^2$$

Where, I is the RGB color vector (the larger d is, the less similar they are), and A is the known pixels number in $\Psi_{p^\wedge}$.

Having found the source exemplar $\Psi_{q^\wedge}$, the value of each pixel to be filled, is copied from its corresponding position.

This suffices to achieve the propagation of both structure and texture information from the source ϕ to the target region Ω, one patch at a time. In fact, we note that any further manipulation of the pixel values (e.g., adding noise, smoothing, etc.) that does not explicitly depend upon statistics of the source region, is more likely to degrade visual similarity between the filled region and the source region, than to improve it.

3. Updating confidence values

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

$$c(q) = c(p^\wedge)\ \forall q \in \Psi_{p^\wedge} \cap \Omega$$

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

As filling proceeds, confidence values decay, indicating that we are less sure of the color values of pixels near the centre of the target region.

3.5 User Interaction

As one of our goals is the automation of the inpainting process, we pay attention to the only phase that the user should be involved with our program, mainly the selection of the region to be inpainted.. This interaction can not be avoided because it is almost impossible for any algorithm to predict which object in the image that the user wants to remove, or in other words, which part of the image he wants to inpaint.

One important part of digital inpainting is the selection of the region to be inpainted Ω. Selection in digital image editing refers to the task of extracting (in some sense) an arbitrary object embedded in an image [35]. It is clear that this is a user interface related problem, and thus, mainly a matter for the human computer-interaction.

The techniques introduced in this thesis do not require the user to specify where the novel information comes from. This is done automatically (and in a fast way), thereby allowing for simultaneously fill-in of multiple regions containing completely different structures and surrounding backgrounds. In addition, no limitations are imposed on the topology of the region to be inpainted.

The only user interaction required by the algorithm is to mask the regions to be inpainted. Since our inpainting algorithm is designed for both restoration of damaged photographs and for removal of undesired objects on the image, the regions to be inpainted must be masked by the user.

It has to be mentioned that creating a suitable mask is essential for obtaining good results, a mask with fairly irregular contours and kinky edges can have an adverse effect on the inpainting process.

In our program, the user can choose between two options to select the target region. In the first option, the program will take in two input images: the original image and another image that would mask out the object. The user can make the mask using any existing software containing object selection features such as Adobe Photoshop and Paint Shop Pro.

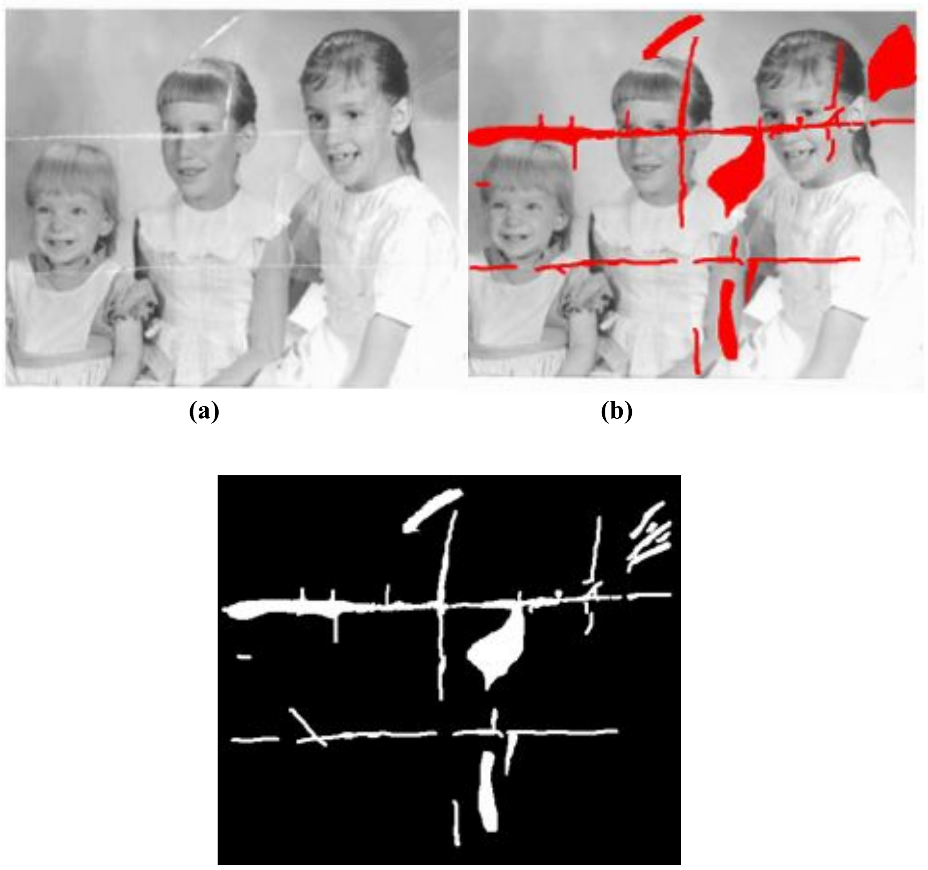
After reading in the mask, we marked the target region that will be filled. We consider the mask as a binary image, where the pixels belong to the target region are set of ones, and the pixels belong to the source region are set of zeros.

This can be expressed mathematically as follows:

$$I(i,j)=0 \qquad \forall I(i,j)\in I-\Omega$$

$$I(i,j)=1 \qquad \forall I(i,j)\in\Omega$$

For example, in figure 3-9, if the user wants to inpaint the thin cracks in the image, he could probably mark these regions with a paint brush tool that exists in many image editing programs. Our program generates a binary mask as shown in figure 3-9 (c), after the user defines which color he used to mark the to-be inpainted region.



(a) (b)

(c)

Figure 3-9 (a) Image for three children. (b) Selection of target region using brush tool. (c) Mask generated using our program.

In the second option, the user displays the original image on screen, and using the mouse, he specifies a polygonal region of interest within the image. Selecting an object via freehand drawing is straightforward. The polygon should be a closed one and it is drawn by clicking a set of points on the image that represent the vertices of the polygon. The enclosed area represents the selection.

Use of normal button clicks adds vertices to the polygon. Pressing Backspace or Delete removes the previously selected vertex. A shiftclick, right-click, or double-click adds a final vertex to the selection and then starts the fill; pressing Return finishes the selection without adding a vertex.

After the selection of the polygon is finished, the inpainting program is executed to fill the region marked by the polygon. The resulted image is displayed, and the user has the ability to repeat the whole process again and again, by selecting another region to inpaint in the image and so on until he gets the result he desires.

For example, in figure 3-10, if the user wants to remove the Sphinx from the image, he can use the mouse to point around the object by clicking on its vertices. The selected area is the target region, and the rest of the image is the source region.
Then the inpainting program encodes the mask by assigning the pixel values inside the polygon to ones, and the pixels outside the polygon are assigned to zeros.

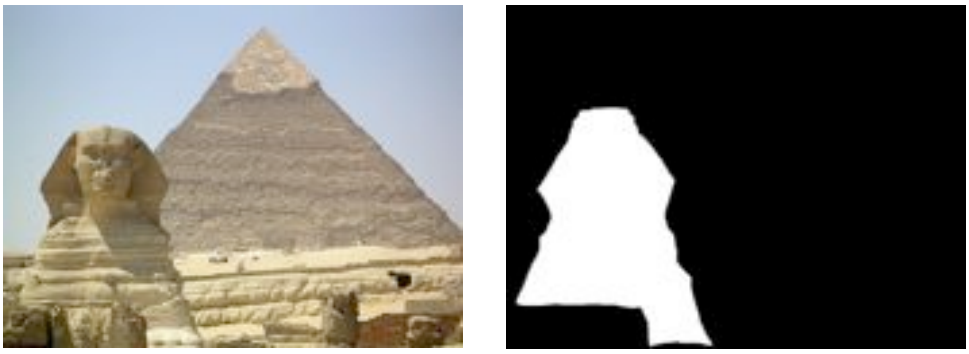


Figure 3-10 (a) The original Image, (b) The mask.

4

Chapter

Results and Comparisons

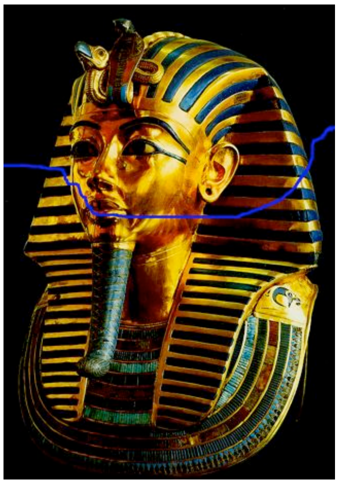
This chapter presents the results of the implemented and developed inpainting algorithms. The algorithms have been tested for a variety of different cases. The results include cases where the algorithms perform well and others where the algorithms fail. The original image, the used mask and the corresponding result are shown for each case. Also, other data such as the computation time, the mask size, i.e., the total number of pixels and the total occluded percentage of the image are presented. The developed inpainting systems are implemented using a C++ code running on a Pentium IV class PC (512 Mb RAM, 2 GHz).

4.1 Restoration Results

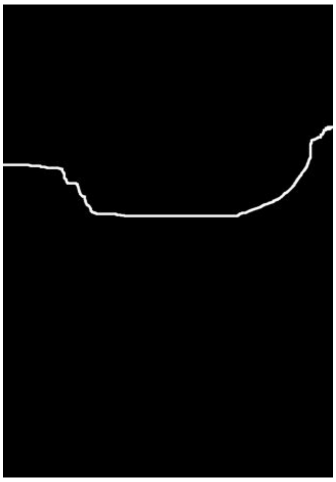
In this section we present the results of inpainting images of varying degree of difficulty. The test images include natural scenes and full color photographs of complex textures.

In the case shown in figure 4-1, we add an artificial thin crack using Photoshop to the image of the famous golden mask of King TUT, The crack is then removed by using the PDE-based digital inpainting algorithm. The image size is 645 by 925, the number of recovered pixels is 4240, and it takes 58 seconds to get this result. Further inpainting iterations would not enhance the image anymore, meaning the PDE algorithm achieves a stable solution. Every ten iterations of inpainting scheme, the target region goes into 3 iterations of anisotropic diffusion, where the time step equals 0.1. We can realize here that the broken lines are entirely removed, but if the cracks are thicker a blurring effect could be introduced to the image.

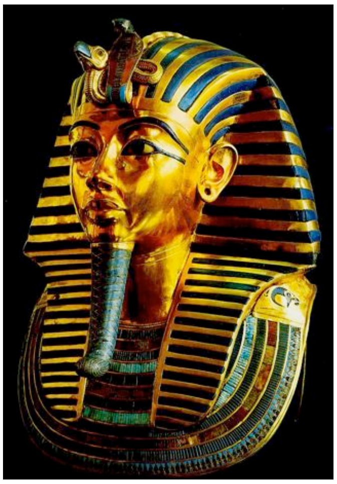
Since the image presented in figure 4-1 is colored one, it is treated as a set of three images corresponding to the R,G,B color channels, and the PDE algorithm is applied sequentially to each one.



(a)



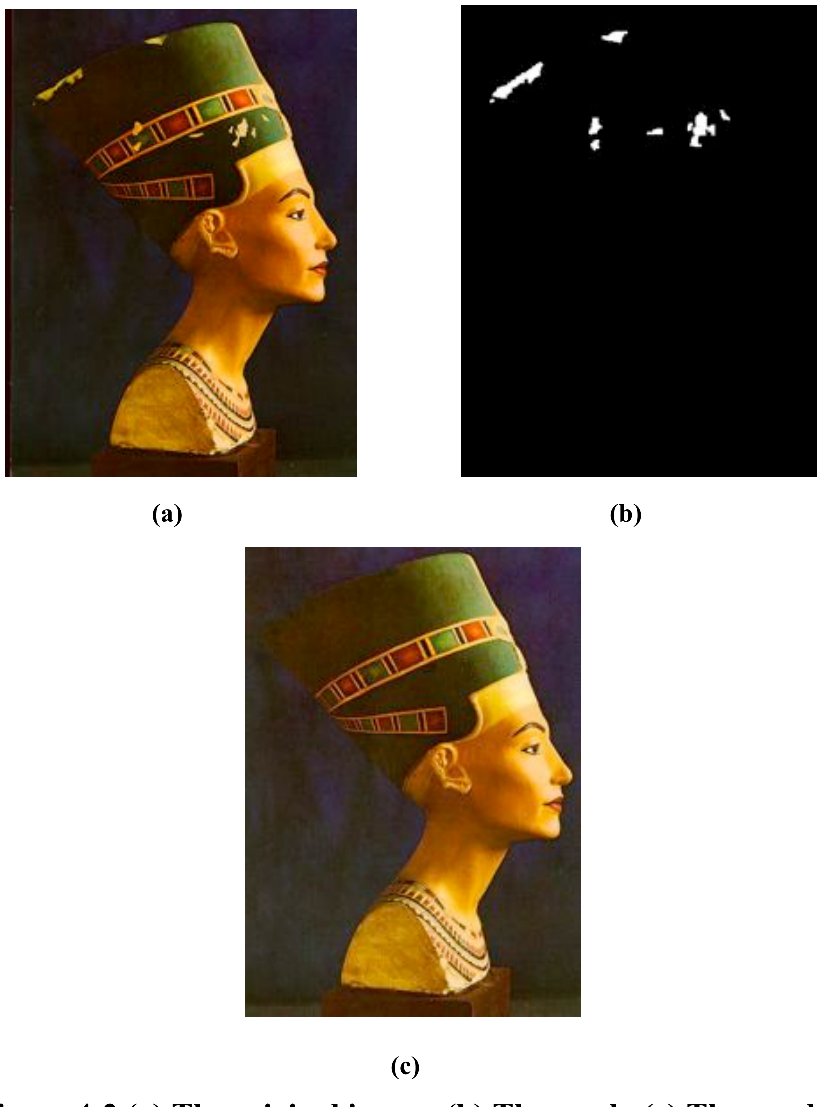
(b)



(c)

Figure 4-1 (a) The original image, (b) The mask, (c) The result.

Figure 4-2 shows the famous bust Statue of Queen Nefertiti before and after inpainting the damaged parts in her crown. This result is achieved by applying the Exemplar-based inpainting algorithm. The image size is 376 by 525; the number of recovered pixels is 3024, and it takes 114 seconds to achieve this result. The patch size used is 9x9 pixels.

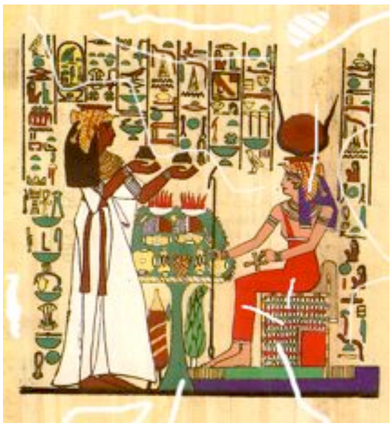
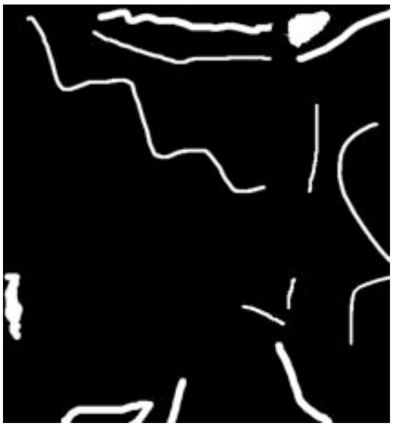


(a) **(b)**

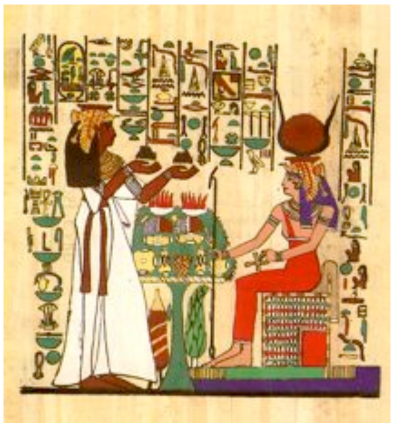
(c)

Figure 4-2 (a) The original image, (b) The mask, (c) The result.

Figure 4-3 displays a vandalized and restored ancient Egyptian picture from the tomb of Nefertari in Upper Egypt. Shown also is the mask used in the inpainting process. The restored image is obtained by using the exemplar-based inpainting algorithm. The image size is 427 by 463 pixels; the number of recovered pixels is 10559, and it takes 292 seconds to get this result. The patch size used is again 9x9 pixels.

(a) **(b)**



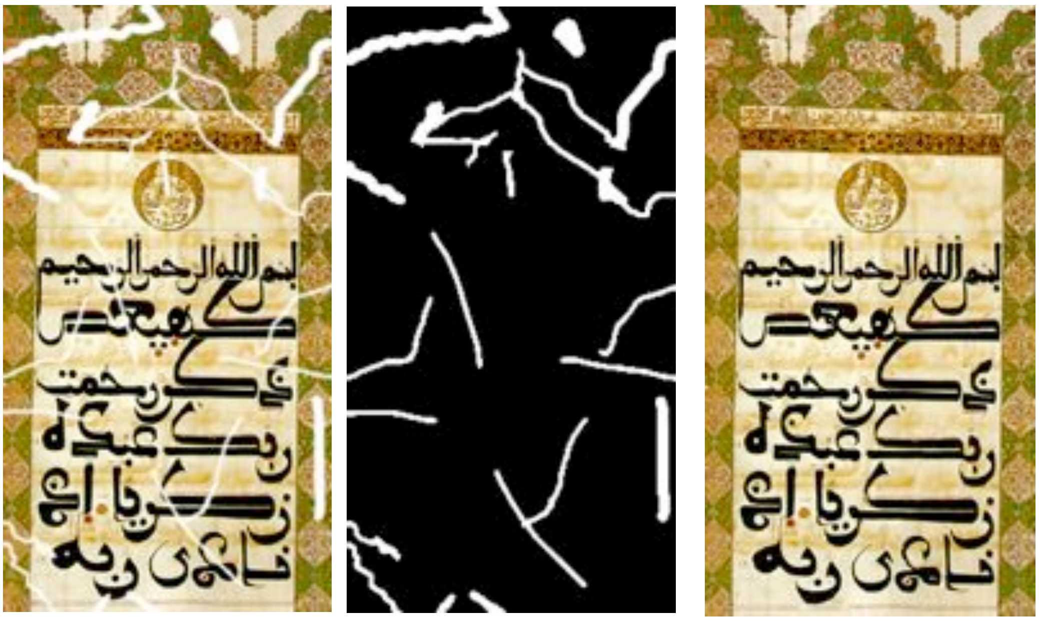
(c)

Figure 4-3 (a) The original image, (b) The mask, (c) The result.

In such cases, where the damaged image contains a fair amount of texture, it is more suitable to use the exemplar-based inpainting algorithm, because the PDE-based inpainting algorithm is not suited for texture reproduction.

Figure 4-4 shows a damaged page of an old Koran manuscript and its restoration. We also use the exemplar-based inpainting algorithm in this case. The image size is 196 by 362; the number of recovered pixels is 6495, and it takes 256 seconds to achieved this result.

From these results, we can easily observe that the speed of the algorithm does not depend only on the size of the to-be inpainted region, but also depends on a very important factor which is the size of the whole image. This is because the exemplar-based algorithm performs an exhaustive search in the whole image in order to find the most similar patch in the source region to the target patch.

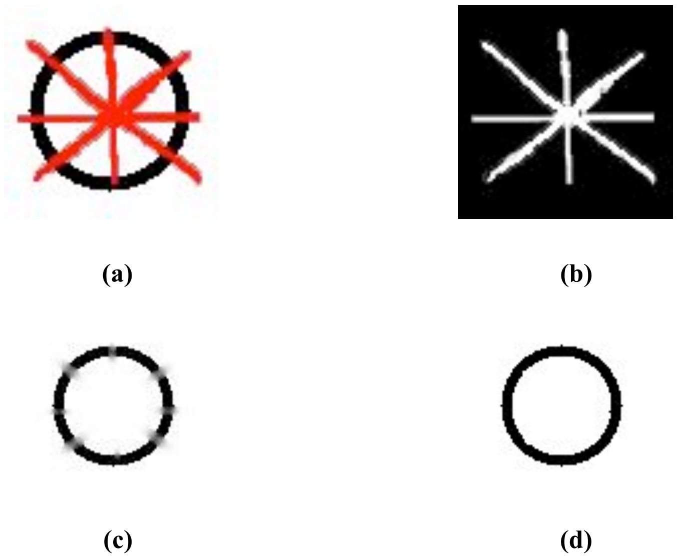


(a) (b) (c)

Figure 4-4 (a) The original image, (b) The mask, (c) The result.

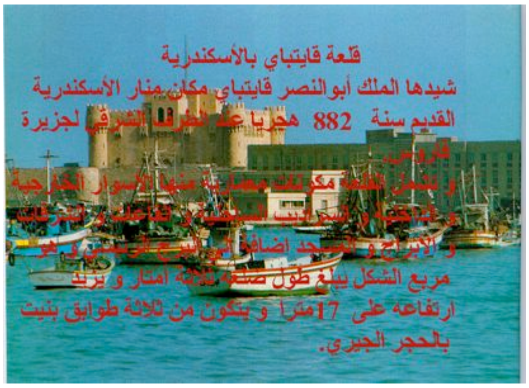
To test the effectiveness of the SDGD algorithm in cases images of pure geometric structure, we implement it for the image of a uniform circle having gaps in its perimeter. The image and the mask defining the region to be inpainted is shown in figure 4-5 (a) and (b). The result shown in figure 4-5 (c) is obtained by Sapiro algorithm [2], and show excessive diffusion in the inpainted areas of the image. In contrast, the results shown in figure 4-5 (d) obtained by using our SDGD-based algorithm show a clean and uniform circle perimeter without any excessive diffusion.

The image size for this case is 80 by 80; the number of recovered pixels is 829, and it takes about 3 seconds by our algorithm to obtain this result.

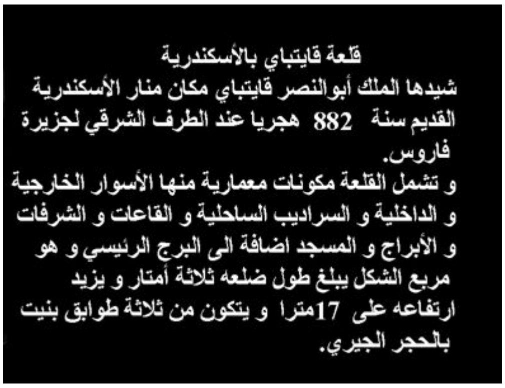

(a) (b)

(c) (d)

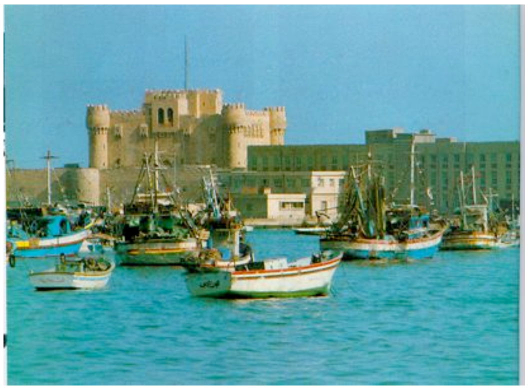
Figure 4-5 (a) Synthetic image, (b) The mask, (c) Result using [2], (d) Result using SDGD algorithm.



(a)


قلعة قايتباي بالأسكندرية
شيدها الملك أبوالنصر قايتباي مكان منار الأسكندرية
القديم سنة 882 هجريا عند الطرف الشرقي لجزيرة
فاروس.
و تشمل القلعة مكونات معمارية منها الأسوار الخارجية
و الداخلية و السراديب الساحلية و القاعات و الشرفات
و الأبراج و المسجد اضافة الى البرج الرئيسي و هو
مربع الشكل يبلغ طول ضلعه ثلاثة أمتار و يزيد
ارتفاعه على 17مترا و يتكون من ثلاثة طوابق بنيت
بالحجر الجيري.

(b)



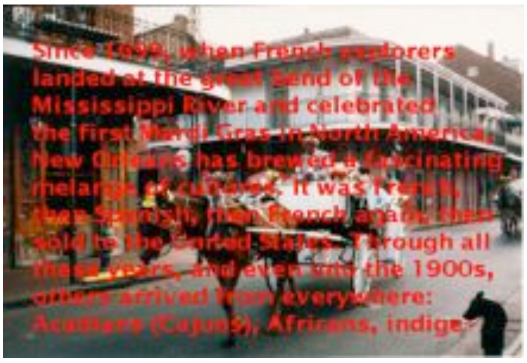
(c)

Figure 4-6 (a) The original image, (b) The mask, (c) The result.

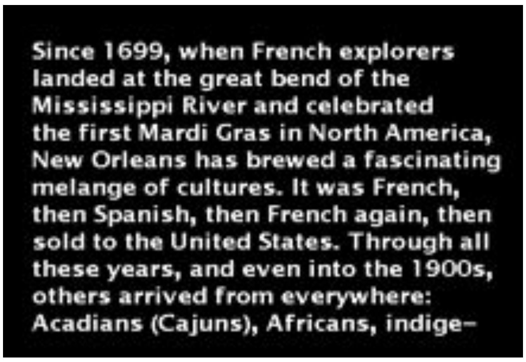
The case shown in Figure 4-6 is example of removing superimposed Arabic text from an image depicting the historic Qaitbay Citadel in Alexandria. Although this image is full of texture, we use the PDE-based inpainting algorithm. This is because the font size of the super imposed text is relatively small. The text here is treated as thin cracks. The image size is 824 by 599; the number of recovered pixels is 45556, and it takes 88 seconds to recover the original image.

The evolution of the inpainting results is displayed in figures 4-6 (c) after 100 iterations. Further inpainting iterations would not enhance the image anymore, meaning that the PDE algorithm achieves a converged solution. Three iterations of anisotropic diffusion are implemented after every ten iterations of the inpainting algorithm.

Two factors affect the speed of the PDE-based algorithm, namely, the size of the region to-be inpainted, and the time step size. Using numerical experimentation, a value of 0.1 is found satisfactory to get the right balance of speed and accuracy.

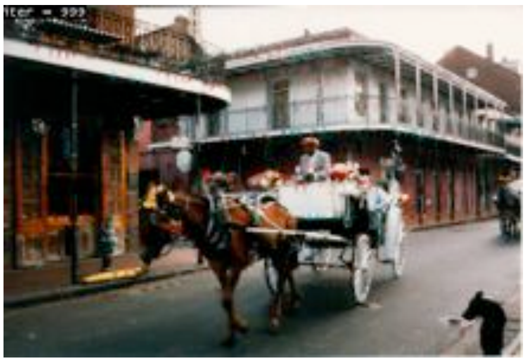


(a)



Since 1699, when French explorers
landed at the great bend of the
Mississippi River and celebrated
the first Mardi Gras in North America,
New Orleans has brewed a fascinating
melange of cultures. It was French,
then Spanish, then French again, then
sold to the United States. Through all
these years, and even into the 1900s,
others arrived from everywhere:
Acadians (Cajuns), Africans, indige-

(b)

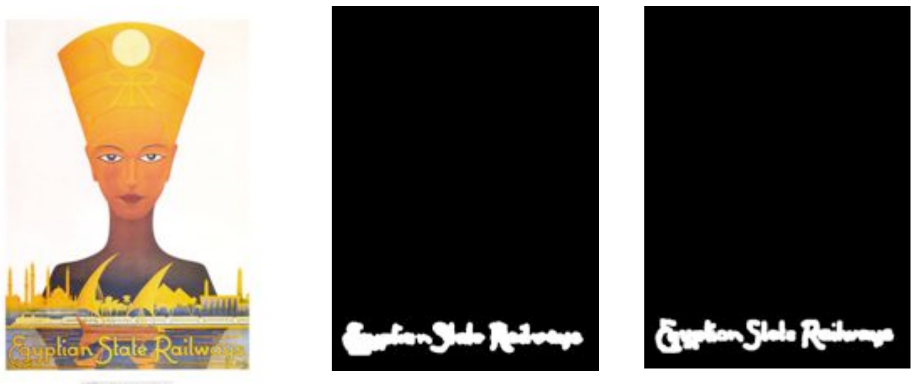


(c)

Figure 4-7 (a) The original image, (b) A mask, (c) The result.

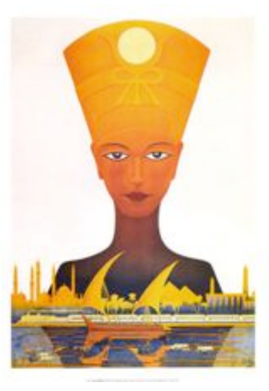
More results of removing superimposed text is shown in figure 4-7, which displays the solution for a standard case in the inpainting literature, depicting a scene in the American city of New Orleans.

It should be noted here, that exact masking of superimposed text may be difficult to achieve, especially for small fonts. Moreover, enlarging the masked region may produce inaccurate results, and increase computational cost. However, our developed SDGD inpainting algorithm is robust enough to handle irregularly masked text. Figure 4-8 shows the results obtained using two different masks that approximately trace the borders of superimposed text. The results indicate that the SDGD inpainting algorithm has successively removed the superimposed text produced identical correct results irrespective of the shape of the masked text letters.



(a) **(b)** **(c)**

Figure 4-8 The original image, and two different masks.


(d)

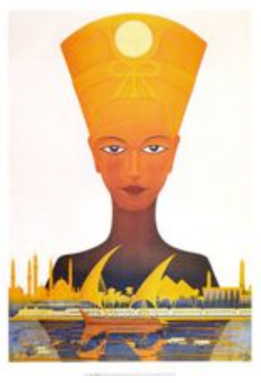
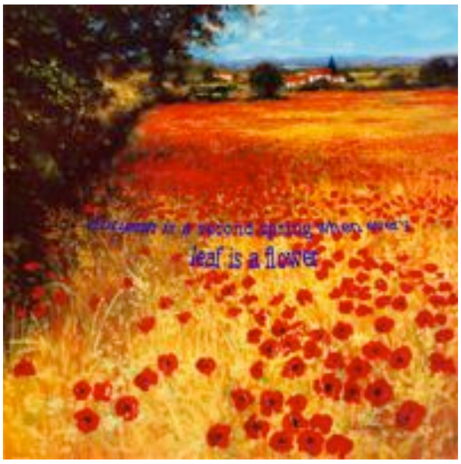
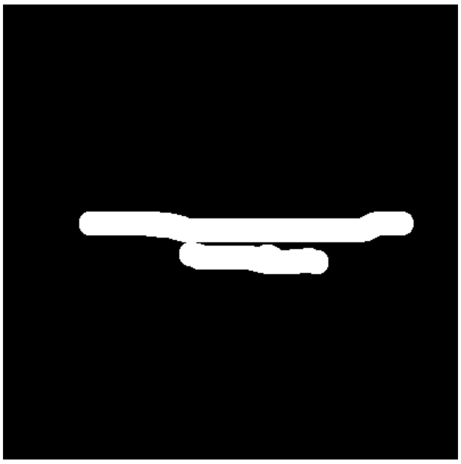

(e)

Figure 4-8 The results using the mask (b), and (c)

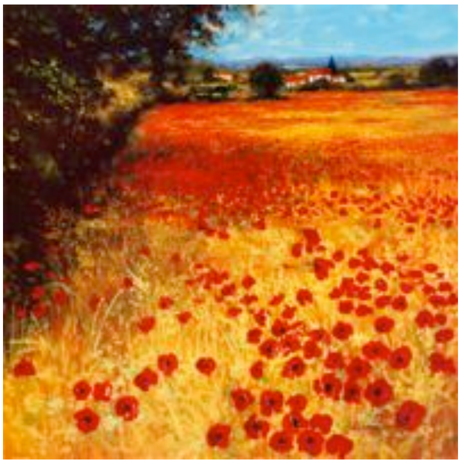
Another case of removing unwanted text is shown in figure 4-9.
In this case the mask doesn’t trace the borders of the text letters at all, rather, a whole region masks the area around the letters. Still, the SDGD inpainting algorithm removed the text, while preserving the texture present in the image.



(a)



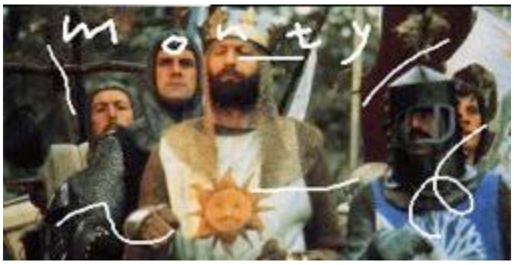
(b)

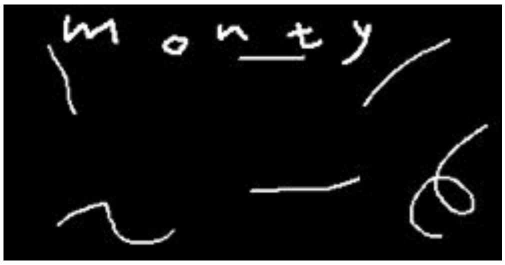


(c)

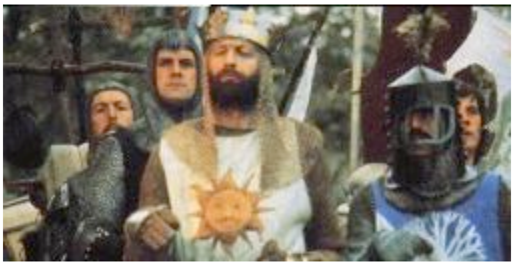
Figure 4-9 (a) The image, (b) A developed mask, (c) The result.

More results of test cases in the inapinting literature are displayed in Figure 4-10 and 4-11 showing the restoration of vandalized and damaged old photos.



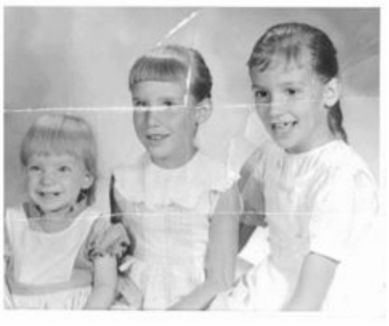


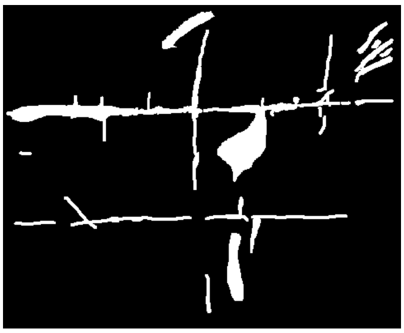
(a) **(b)**



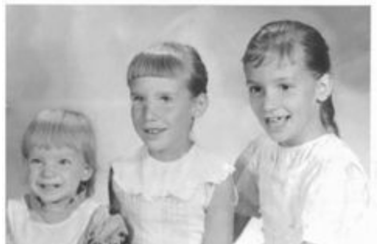
(c)

Figure 4-10 (a) The image, (b) A mask, (c) The result.





(a) **(b)**



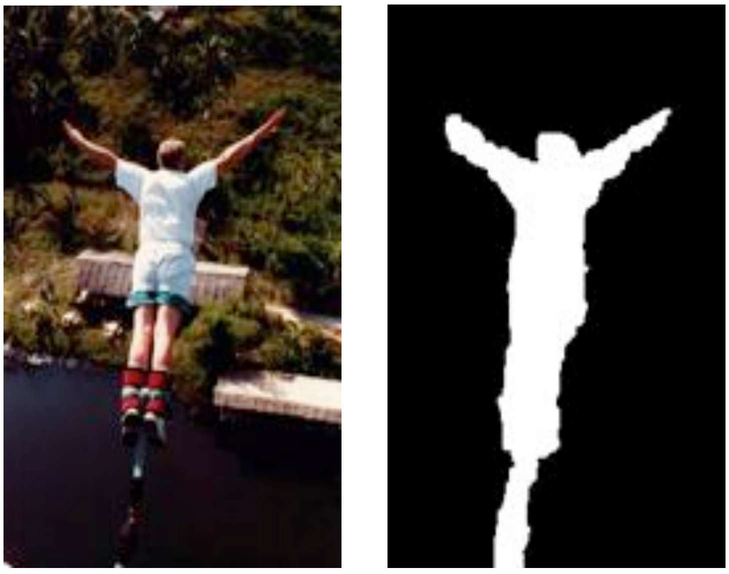
(c)

Figure 4-11 (a) The image, (b) A mask, (c) The result.

4.2 Object Removal Results

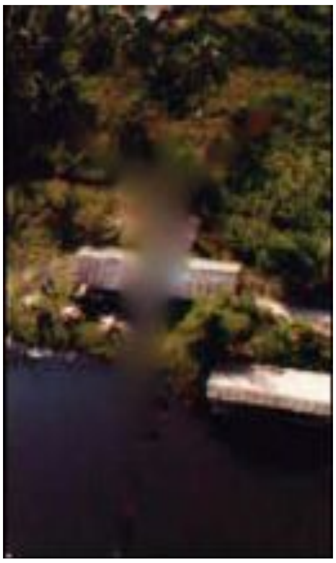
In this section we present the results of object removal from images. Such applications are usually characterized by texture dominated images and removal of large areas in the image**.**

In figure 4-12, we compare the algorithms of reference [2], [31], and our algorithm presented in this thesis. The result of the algorithm used in reference [2] shows a marked image blurring and a loss of details as shown in figure 4-12(c). While that of reference [31] shown in figure 4-12(e) exhibits a break in the linear structure of (the white building) in the image. Better image texture and structure are evident in the result of our algorithm that includes either gradient or SDGD information the in the present algorithm as shown in figure 4-12(e) and (f). The image size for this case is 206 by 308, and the number of recovered pixels is 7996.

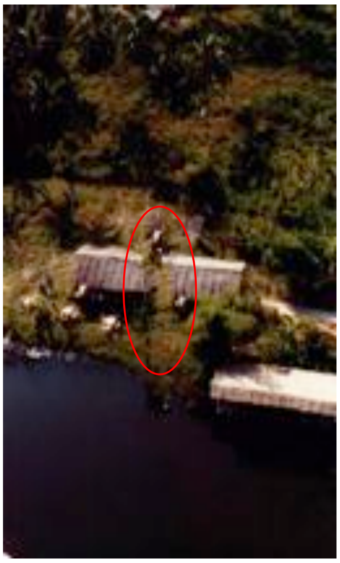


(a) (b)

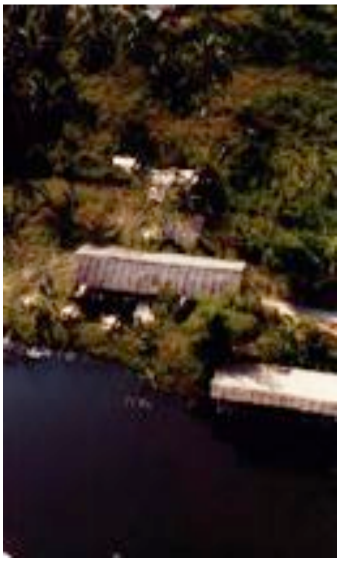
Figure 4-12 (a) The original image, and (b) The mask



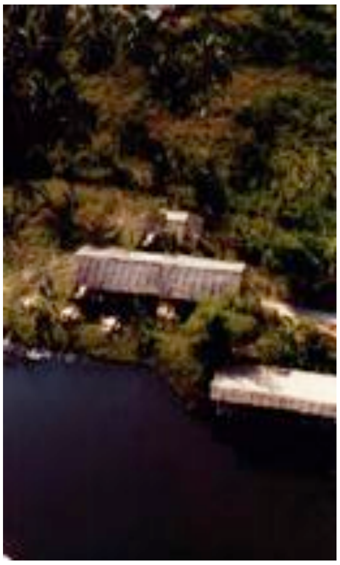
(c)



(d)



(e)



(f)

Figure 4-12 (c) Result of [2], (d) Result of [31], (e) Result of Gradient algorithm, (f) Result of SDGD algorithm.

This comparison indicates as expected that the PDE-based digital inpainting algorithm has some disadvantages, which can be summarized as follows:

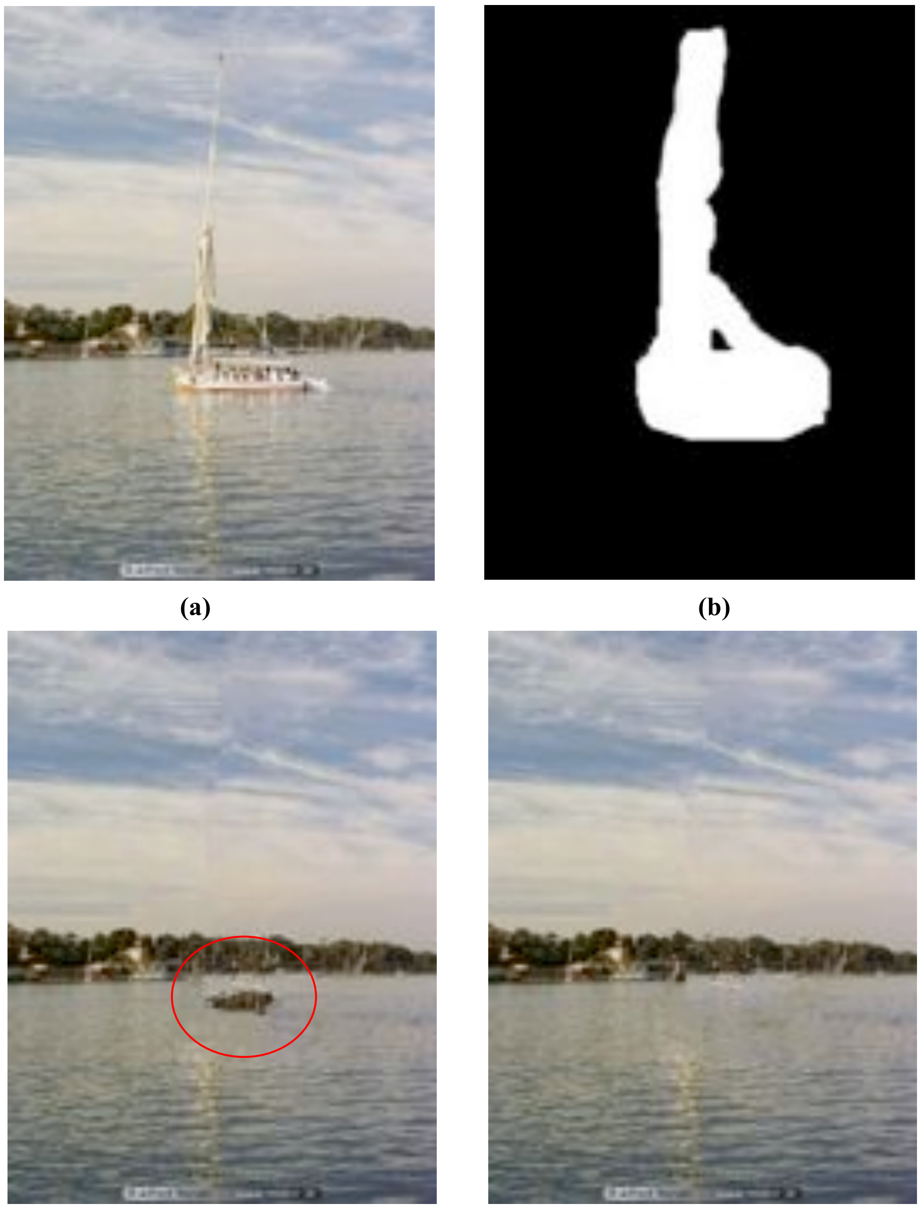
- Resulting image is blurred.
- Large textured regions are not well reproduced.

Also the texture Exemplar-based inpainting algorithm fails in some cases, and that is because:

- The algorithm can not accurately propagate image structures.
- The matching criterion for texture synthesis that only uses only the color information produced artifacts in the image (garbage growing)

The algorithms developed in this thesis eliminate most of these drawbacks and produced better results. Our main contribution is the use of using the gradient SDGD information to calculate the distance function and patch priorities during the propagation of structure and texture information.

Another comparison is presented in Figure 4-13 which displays an image for a sail boat in the Nile, the mask used to remove the sail boat from the image, and the inpainted image. Figure 4.13 (c) shows the inpainted image using Criminisi algorithm [32]. The result of our inpainting algorithm is shown in figure 4.13 (d). These results demonstrate the effectiveness of our method in eliminating garbage growing that could not be avoided by the method of [32]. The size of this image is 200 by 267, and the number of recovered pixels is 7537, and the run time is 73 seconds.

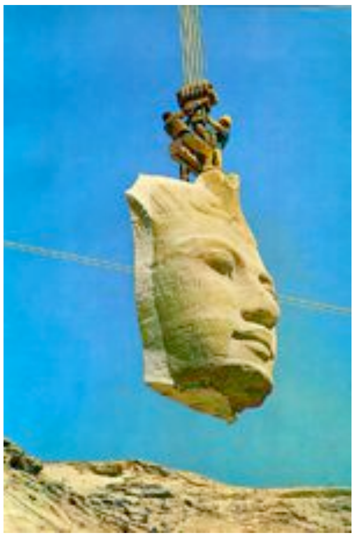


(a) (b)

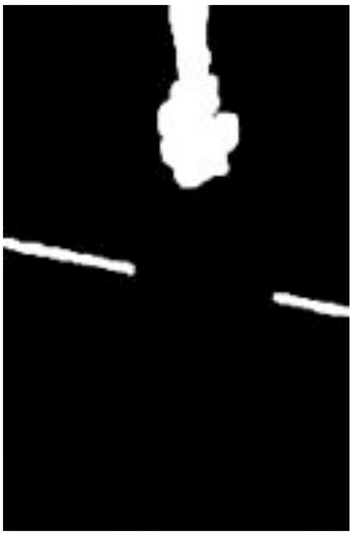
(c) (d)

Figure 4-13 (a) The original image, (b) The developed mask, (c) The result using algorithm of [32], (d) The result using method presented in section 3.1.

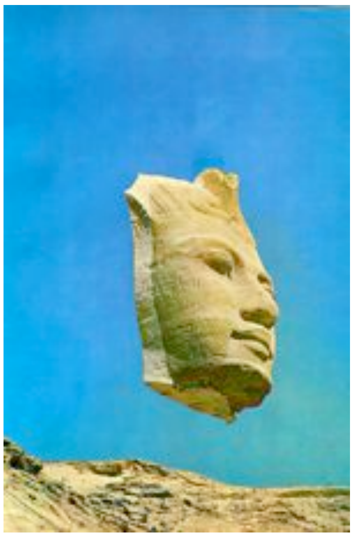
More results are presented in Figures 4-14 and 4-15. Figure 4-14 displays an archival picture during the removal of the Abu Sembel Temple to a new location during the construction of the High Dam in Aswan in southern Egypt. The steel cables and the workers appearing in the left image have been removed as shown.



(a)



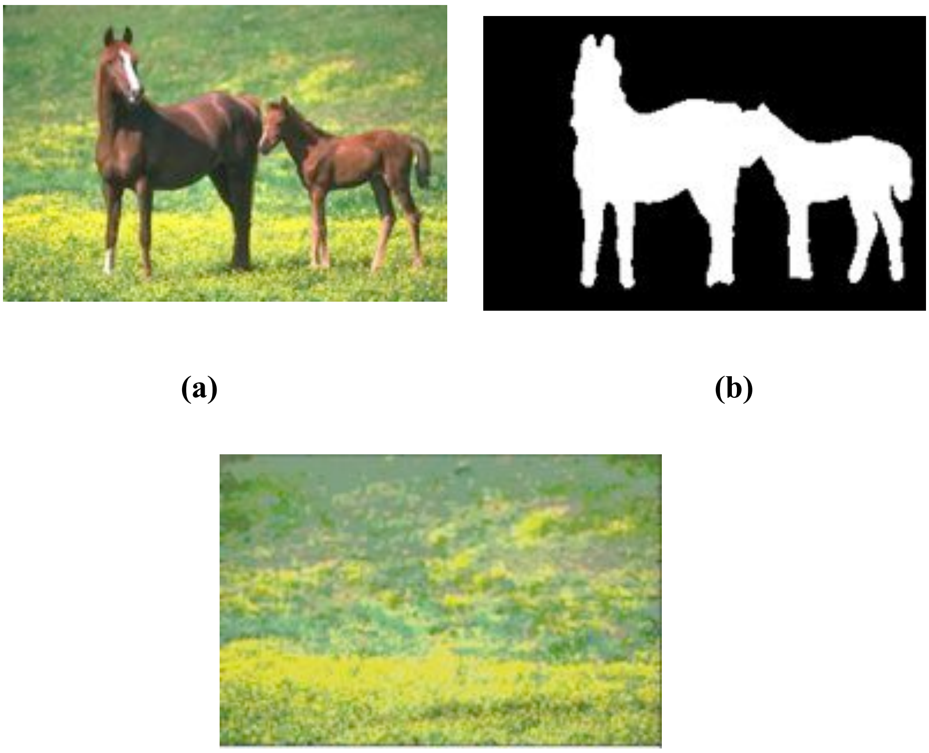
(b)



(c)

Figure 4-14 (a) The original image, (b) The developed mask, (c) The result

Figure 4-15 shows the effectiveness of our modified distant function inpainting algorithm in removing large regions in a natural scene image, and still preserves the integrity of the image. The size of this image is 240 by 160, and the number of recovered pixels is 11196.

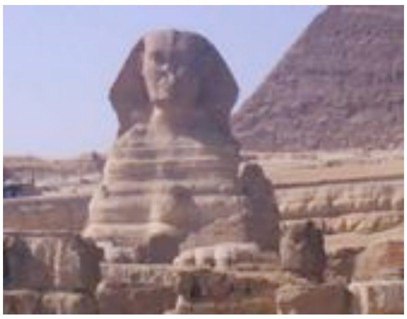


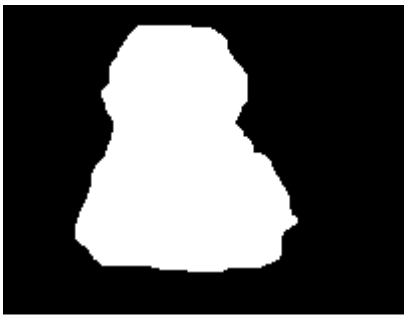
(a) (b)

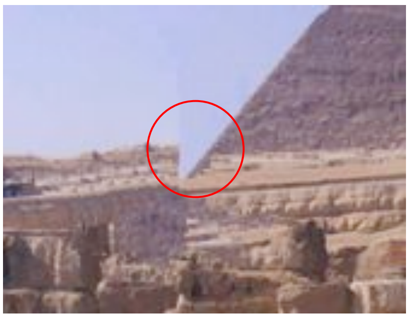
(c)

Figure 4-15 (a) The original image, (b) The mask, (c) The result.

In some cases, removing an object from the image is a difficult task. This occurs when the texture of the object to be removed differes slightly from the rest of the image texture. Figure 4-16 (a) shows a view for the sphinx with the pyramid in the background, and algorithm is implemented to remove the sphinx from the image. Figure 4-16 (b) depicts the mask used to remove the sphinx from the image. The inpainting process for this image is quite difficult, since the textures present in the image are very close to each other (the pyramid and the sphinx), the result using the exemplar algorithm [32] is shown in figure 4-16 (c), and finally the result using our enhanced distance function inpainting method is shown in figure 4-16 (d).


(a)


(b)


(c)

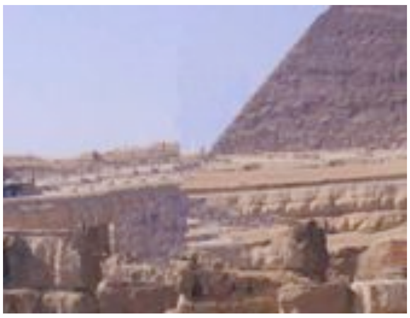

(d)

Figure 4-16 (a) The original image, (b) A developed mask, (c) The result using [32], (d) The result using present algorithm.

A similar case is presented in figure 4-17, where a large region (the sphinx) is required to be removed and to recover the occluded part of the pyramid. The difficulty in this case is due to the similarity between the texture of the pyramid, the sphinx, the sand, and the rocks. Additionally, the pyramid has a linear structure and the occluding object (the sphinx) has a curved structure. The result of reference [32] algorithm is displayed in figure 4-17 (c) and exhibits spurious artifacts in the inpainted image. A satisfactory result is achieved using our algorithm as shown in figure 4-17 (d).

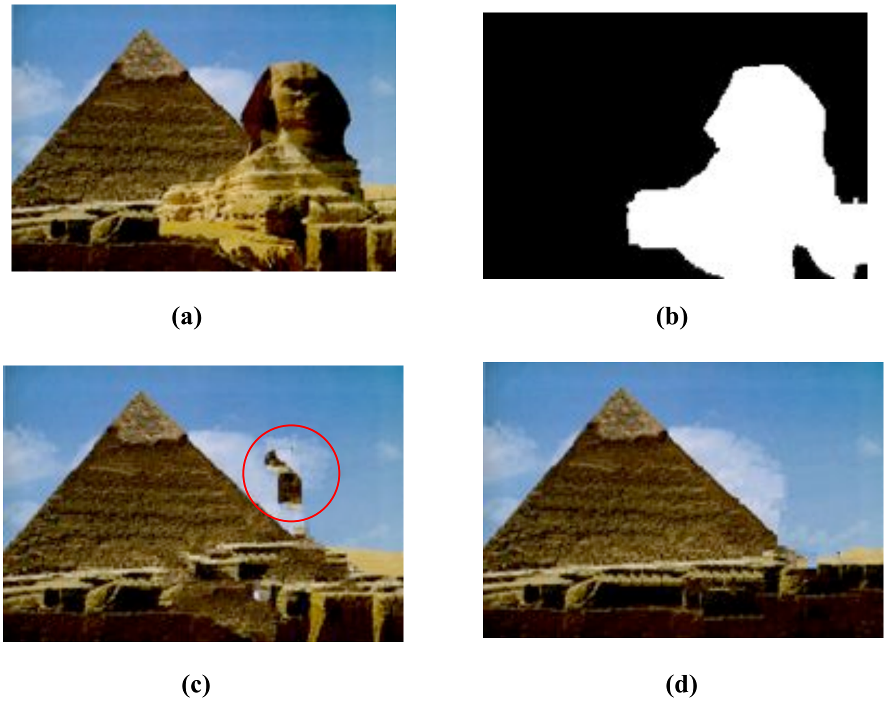


Figure 4-17 (a) The original image, (b) A developed mask, (c) The result using [31], (d) The result using SDGD algorithm.

Figure 4-18 displays an image for a man sitting on a curved rock. This test case is reported in [6], and the author shows how the exemplar-based inpainting algorithm developed by Criminisi [31] fails to decide on the right patch to use during the inpainting process, and produces artifacts as shown in figure 4-18 (c). Better result is achieved using our developed algorithms as shown in figure 4-18 (d).

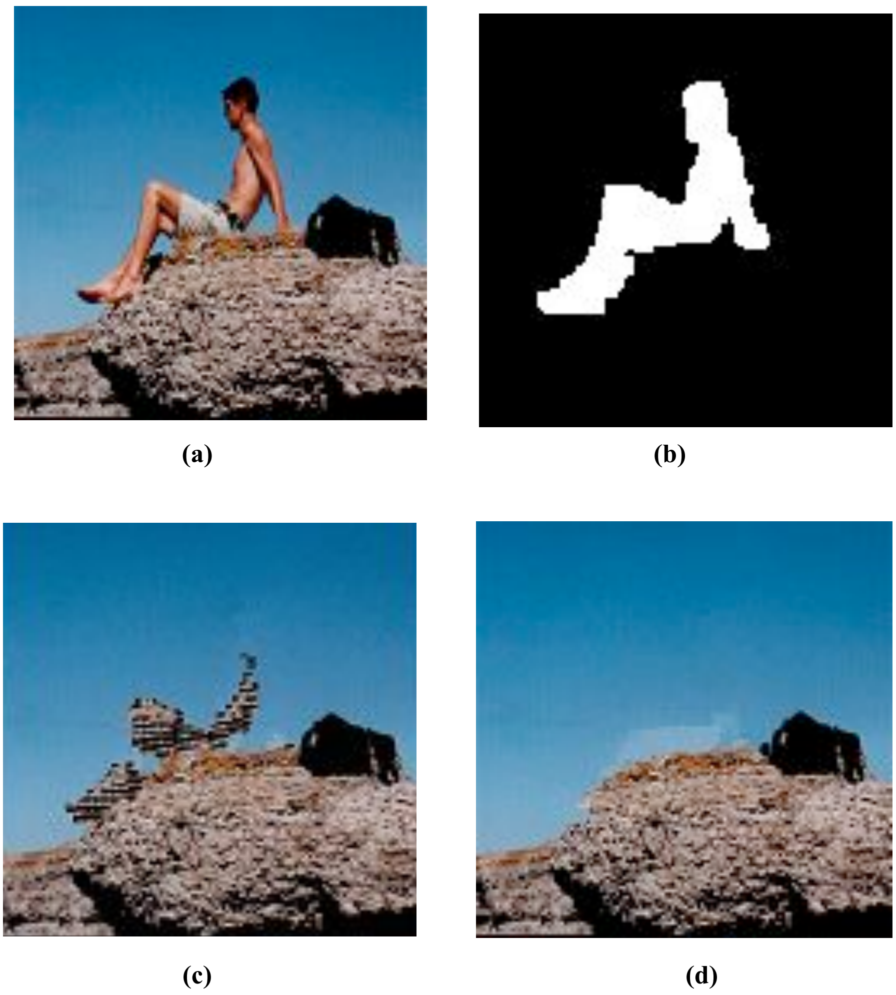


Figure 4-18 (a) The original image, (b) A mask, (c) The result using [31], (d) The result using SDGD algorithm.

4.3 Inpainting Algorithms Parameters

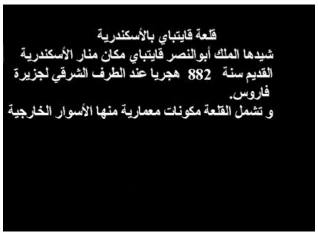
In this section, we discuss the factors affecting the speed of the inpainting algorithms. The two main parameters are the size of the region to be inpainted and the patch size. This discussion will help us to evaluate our proposed inpainting algorithm and to compare it with the developed inpainting algorithms presented earlier in chapter 3.

4.3.1 Mask Size

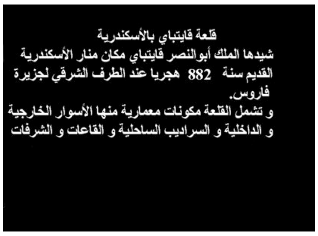
The first factor we experiment with is the size of the inpainted region. As a test case for the PDE-based inpainting algorithm, we chose the Quaitbay citadel image (figure 4-6). We executed the algorithm using the different mask sizes shown in figure 4-19.

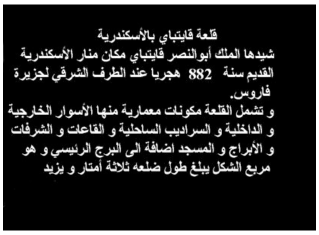
قلعة قايتباي بالأسكندرية

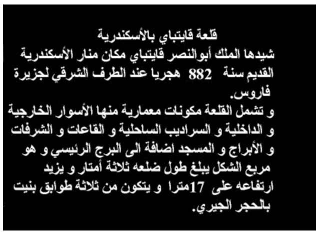
قلعة قايتباي بالأسكندرية
شيدها الملك أبوالنصر قايتباي مكان منار الأسكندرية
القديم سنة 882 هجريا عند الطرف الشرقي لجزيرة


قلعة قايتباي بالأسكندرية
شيدها الملك أبوالنصر قايتباي مكان منار الأسكندرية
القديم سنة 882 هجريا عند الطرف الشرقي لجزيرة
فاروس.
و تشمل القلعة مكونات معمارية منها الأسوار الخارجية

(a) (b) (c)


قلعة قايتباي بالأسكندرية
شيدها الملك أبوالنصر قايتباي مكان منار الأسكندرية
القديم سنة 882 هجريا عند الطرف الشرقي لجزيرة
فاروس.
و تشمل القلعة مكونات معمارية منها الأسوار الخارجية
و الداخلية و السراديب الساحلية و القاعات و الشرفات


قلعة قايتباي بالأسكندرية
شيدها الملك أبوالنصر قايتباي مكان منار الأسكندرية
القديم سنة 882 هجريا عند الطرف الشرقي لجزيرة
فاروس.
و تشمل القلعة مكونات معمارية منها الأسوار الخارجية
و الداخلية و السراديب الساحلية و القاعات و الشرفات
و الأبراج و المسجد اضافة الى البرج الرئيسي و هو
مربع الشكل يبلغ طول ضلعه ثلاثة أمتار و يزيد


قلعة قايتباي بالأسكندرية
شيدها الملك أبوالنصر قايتباي مكان منار الأسكندرية
القديم سنة 882 هجريا عند الطرف الشرقي لجزيرة
فاروس.
و تشمل القلعة مكونات معمارية منها الأسوار الخارجية
و الداخلية و السراديب الساحلية و القاعات و الشرفات
و الأبراج و المسجد اضافة الى البرج الرئيسي و هو
مربع الشكل يبلغ طول ضلعه ثلاثة أمتار و يزيد
ارتفاعه على 17مترا و يتكون من ثلاثة طوابق بنيت
بالحجر الجيري.

(d) (e) (f)

Figure 4-19 (a)-(f) Different masks for the Quaitbay Citadel image.

As expected, we find that the larger the size of the mask, the more time needed by the algorithm to achieve satisfying results. The relationship between the number of recovered pixels and the time taken by the PDE-based inpainting algorithm is shown in figure 4-20. From this graph we conclude that the relationship between the mask size and the speed of the algorithm is linear.

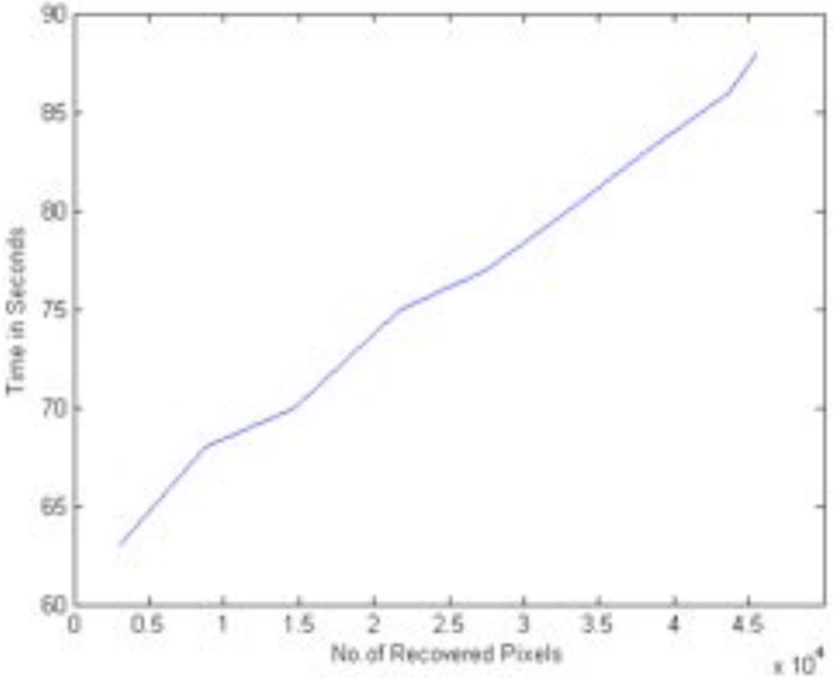


Figure 4-20 PDE algorithm evaluation graph.

The same procedure is implemented for the texture-based inpainting algorithm; we used the different masks shown in figure 4-21. We implemented the algorithm for the image of Nefertari's Tomb (figure 6-3), and the result also exhibits a linear relationship as shown in figure 4-22

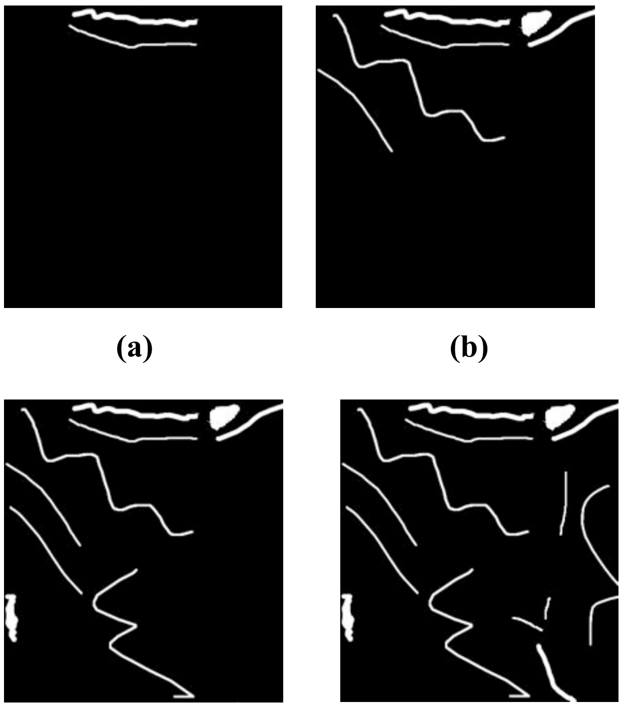


Figure 4-21. Different masks of Nefertari' Tomb image.

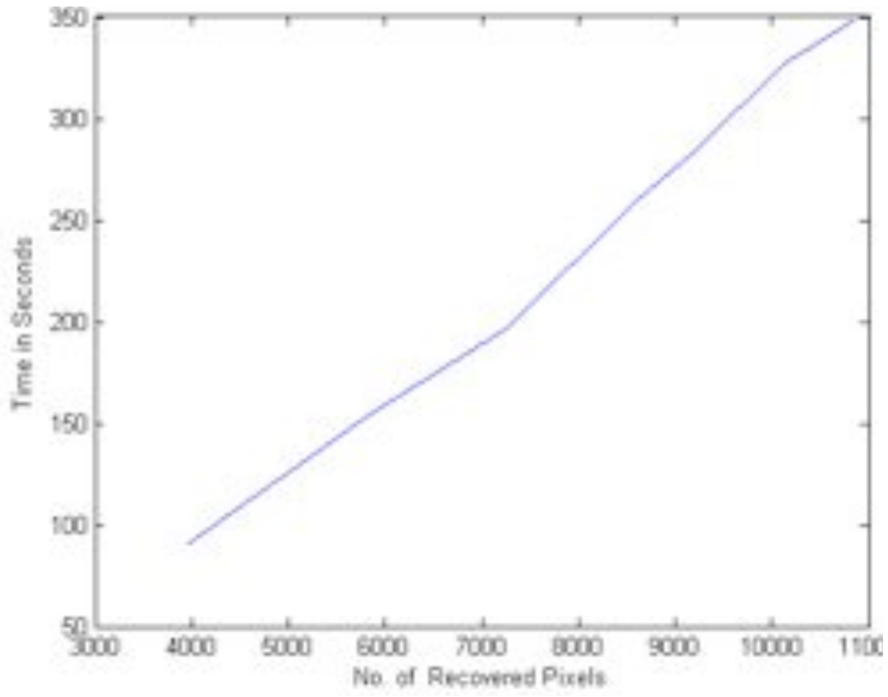


Figure 4-22 Texture-based algorithm evaluation graph

4.3.2 Patch Size

The second important factor that affects the performance of any texture based inpainting algorithm is the patch size. However, it is difficult to set a universal patch size that can be applied to all images. Using large patch size, the filling rate is high, which leads to faster execution time of the inpainting algorithm. However, there are more important implications on choosing the right patch size.

As stated in reference [31], the patch should be slightly larger than the largest distinguishable texture element. If the patch size is too small, it will have little or no texture characteristics which result in the production of a mass of small fragments. On the other hand, if it is too large, the patch loses the local texture details, which will lead to a mismatch. The proper size of the patch as given in [31] is 9x9 pixels. Experimenting with the patch size however, indicate that the proper patch size can vary from one image to another. The results shown in figure 4-23 show the effect of changing the patch size on the quality of the inpainted image. Our experimentation indicates that in general, a patch size between 7x7 and 11x11 pixels should be appropriate for most images.

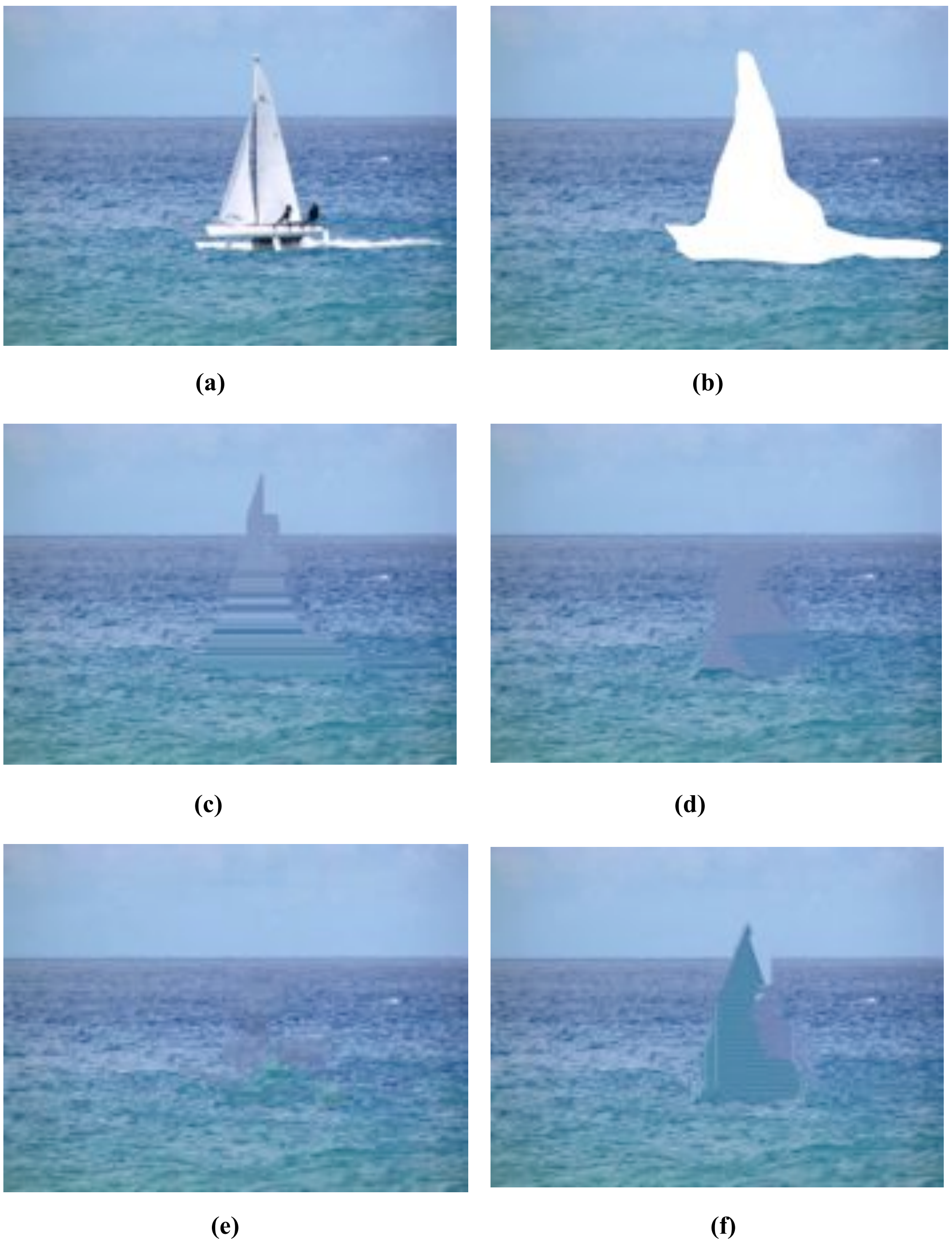


Figure 4-23 (a) Sail boat image, (b) The mask, (c) Patch size=5x5, (d) Patch size=7x7, (e) Patch size=9x9, and (f) Patch size=11x11.

5

Chapter

Conclusions and Future Work

5.1 Conclusions

In this thesis we have developed an improved inpainting method combining PDE-based and texture-based algorithms. The choice of using Texture or PDE algorithms depends on the nature of the image to be inpainted. The PDE algorithm is used for structure dominated images to fill-in narrow or crack type regions, while the texture algorithm is more suited for textured images. The time required for the inpainting process depends on the size of the image and the regions to be inpainted, and it ranges form few seconds to several minutes for large images.

Our main contribution in the enhanced inpainting method includes:

1. Modifying the patch filling order scheme by setting the data term to include second derivative information in the gradient direction (SDGD).
2. Modifying the distance metric function to include the image gradient information in addition to intensity values.

Several test images have been used and the results demonstrate that our developed algorithm can reproduce texture and at the same time keep the structure of the surrounding area of the inpainted region.
Our method proved to be effective in removing large objects from an image, ensuring accurate propagation of linear structures, and eliminating the drawbacks of "garbage growing" and image blurring which are common problems in other methods. The results obtained are preferable to those obtained by other similar methods.

The results presented in the thesis demonstrate the effectiveness of the inpainting method for several test cases involving the restoration of old and damaged pictures or manuscripts, and the removal of superimposed text and large objects.

We hope that the findings of this thesis may help to conserve and enhance old manuscripts and other cultural treasures which otherwise would be lost to decay.

5.2 Future Work

Future work is planned in two areas:

3D Extension

We aim to extend the presented methodology to three dimensions for applications involving damaged monuments and historical artifacts.

Decreasing user intervention

The inpainting system should be able to automatically switch between PDE-based algorithm and Texture-based algorithm according to the nature of the image. As already pointed out in [40] each separate step of the inpainting algorithm could be performed with several different sub-algorithms. Since it is unlikely that one combination performs optimally, it would be desirable to have a criterion for automatically choosing the appropriate algorithms. This criterion would have to include the form of the inpainting domain, image contents, and amount of texture.

References

1. Zinovi Tauber, "**Disocclusion and Photorealistic Object Reconstruction and Reprojection"**, Ph. D. Thesis, Simon Fraser University, April 2004.
2. M. Bertalmio, G. Sapiro, V. Caselles and C. Ballester, **"Image Inpainting"**, Proceedings of SIGGRAPH 2000, New Orleans, USA, pp 417-424, July 2000.
3. M. Bertalmio, A.L. Bertozzi, and G. Sapiro, **"Navier stokes, Fluid Dynamics, and Image and Video Inpainting"**, Proceedings of Conf. Comp.Vision Pattern Rec., Hawai, pp.355–362, Dec 2001.
4. J. Shen, **"Inpainting and the Fundamental Problem of Image Processing"**, SIAM News 36(5), June 2003.
5. T.F. Chan, J. Shen and L. Vese, **"Variational PDE Models in Image Processing"**, UCLA Computational and Applied Mathematics Reports 02-61, Dec. 2002.
6. Samuel Adolfson, **"Algorithms Regarding Automatic Retouching of User Selected Regions in Digital Images"**, Master's Degree Project, Royal Institute of Technology, Stockholm, Sweden, 2004.
7. K. Patwardhan, G. Sapiro, and M. Bertalmio, **"Video Inpainting of Occluded and Occluding Objects"**, Proceedings of IEEE International Conference on Image Processing, 2005.
8. Y.Wexler, E. Shechtman, and M. Irani, **"Space-time Video Completion"**, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04), vol. 1, pp. 120–127, June 2004.

9. R. Bornard, E. Lecan, L. Laborelli And J-H. Chenot, **“Missing Data Correction in Still Images and Image Sequences”**, ACM Multimedia, 2002.
10. A. RareS, M.J.T. Reinders, J. Biemond And R.L. Lagendijk, **“A Spatio-Temporal Image Sequence Restoration Algorithm”**, The 2002 International Conference on Image Processing (ICIP 2002), Rochester, New York, USA, September 2002.
11. A.C. Kokaram, R.D. Morris, W.J. Fitzgerald And P.J.W. Rayner, **“Interpolation of Missing Data in Image Sequences”**, IEEE Transactions on Image Processing. Vol. 4. no.11, pp 1509-1519, Nov. 1995.
12. J. Verdera, V. Caselles, M. Bertalmio and G. Sapiro, **"Inpainting Surface Holes"**, IEEE International Conference on Image Processing ICIP, Spain, September 2003.
13. J. Davis, S. Marschner,M. Garr and M. Levoy, **“Filling Holes in Complex Surfaces using Volumetric Diffusion”**, Proceedings of First International Symposium on 3D Data Processing, Visualization, Transmission, 2002.
14. S. Masnou, **“Disocclusion: A Variational Approach Using Level Lines”**, IEEE Transactions on Signal Processing, 11(2), p.68–76, February 2002.
15. M. Oliveira, B. Bowen, R. McKenna, Y. Chang, **“Fast Digital Inpainting”**, Proceedings of the International Conference on Visualization, Imaging and Image Processing, Marbella, Spain, pp. 261–266, 2001.
16. C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, J. Verdera, **“Filling–In by Joint Interpolation of Vector Fields and Gray**

Levels", IEEE Transactions on Signal Processing, 10(8) , p.1200–1211, August 2001.

17. S. Esedoglu and J. Shen, **"Digital Inpainting Based on the Mumford-Shah-Euler Image Model"**, European Journal of Applied Mathematics, 2002.
18. A. Bertozzi, S. Esedoglu, and A. Gillette, **"Analysis of a Two-Scale Cahn-Hilliard Model for Image Inpainting"**, Submitted for publication in IEEE Trans. Image Proc., 2006.
19. CELIA A. ZORZO BARCELOS, MARCOS AUR´ELIO BATISTA, **"Image Inpainting and Denoising by Nonlinear Partial Differential Equations",** IEEE Computer Society Conference on Computer Graphics and Image Processing, 2003.
20. H. Yamauchi, J. Haber, H.-P. Seidel, **"Image Restoration Using Multiresolution Texture Synthesis and Image Inpainting",** Computer Graphics International, pp.108-113, 2003.
21. Li-Yi Wei and Marc Levoy, **"Fast Texture Synthesis Using Tree-Structured Vector Quantization"**, In Proceedings of SIGGRAPH 2000.
22. Liang, L., Liu, C., Xu, Y., Gguo, B., and Shum, "**Real-time Texture Synthesis Using Patch-based Sampling",** ACM Trans. Graphics, pp.127–150, 2001.
23. I. Drori, D. Cohen-Or, and H. Yeshurun, **"Fragment-Based Image Completion"**, SIGGRAPH, 2003.
24. P. Pérez, M. Gangnet, and A. Blake, **"PatchWorks: Example-Based Region Tiling for Image Editing"**, Technical Report, Microsoft Research, MSR-TR-2004-04, 2004.

25. V. Kwatra, A. Schodl, I. Essa, G. Turk and A. Bobick, **“Graphcut Textures: Image and Video Synthesis Using Graph Cuts”**, in Proc. SIGGRAPH 2003.
26. M. Ashikhmin, **“Synthesizing Natural Textures”**, Proceedings of ACM Symposium on Interactive 3D Graphics, Research Triangle Park, NorthCarolina, March 2001.
27. BianRu Li, Yue Qi, XuKun Shen, **"An Image Inpainting Method"**, Ninth International Conference on Computer Aided Design and Computer Graphics, IEEE, 2005.
28. Kuei-Yuan Cheng, **“Research on Improving Exemplar-Based Inpainting”**, Ph.D. Thesis, National Taiwan University, Taipei, Taiwan, June 2005.
29. Mag. Harald Grossauer , **"Completion of Images with Missing Data Regions"**, Ph.D. Thesis, University of Innsbruck, Austria, 2005.
30. R.J. Cant and C.S. Langensiepen, **"A Multiscale Method for Automated Inpainting"**, 17th European Simulation Multiconference, 2003.
31. Criminisi A., Perez P., and Toyama K., **"Region filling and Object Removal by Exemplar-based Image Inpainting"**, IEEE Trans. Image Processing 9, pp1200-1212, 2004.
32. A. Criminisi, P. Peres and K. Toyama, **"Object Removal by Exemplar-Based Inpainting"**, Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’03).
33. T. F. Chan and J. Shen. **“Mathematical Models of Local Non-Texture Inpaintings”**, SIAM Journal on Applied Mathematics, 62(3):1019–1043, 2001.

34. R.C. Gonzales and R.E. Woods, **“Digital Image Processing”**, Second Edition, Prentice Hall, Inc. 2002. ISBN: 0-201-18075-8

35. T.F. Chan and J. Shen, **“Non-Texture Inpainting by Curvature Driven Diffusion (CDD)”**, UCLA Computational and Applied Mathematics Reports 00-35, September 2000.

36. P. Perona and J. Malik, **"Scale Space and Edge Detection Using Anisotropic Diffusion"**, IEEE-PAMI 12, pp. 629-639, 1990.

37. L. Rudin, S. Osher, and E. Fatemi, **“Nonlinear Total Variation Based Noise Removal Algorithms”**, Physica D, 60(1-4):259–268, 1992.

38. T. Chan, J. Shen, and S. Kang, **“Euler’s Elastica and Curvature-Based Image Inpainting”**, SIAM Journal on Applied Mathematics, 63(2):564–592, 2002.

39. M. Nitzberg, D. Mumford, and T. Shiota, **“Filtering, Segmentation, and Depth”**, Number 662 in Lecture Notes in Computer Science. Springer-Verlag, 1993.

40. M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, **"Simultaneous Structure and Texture Image Inpainting",** Proceedings of IEEE conference on Computer Vision and Pattern Recognition, 2003.

41. Alexei A. Efros and Thomas K. Leung, **“Texture Synthesis by Non-Parametric Sampling”**, IEEE International Conference on Computer Vision, 1033–1038, 1999.

42. H. Grossauer and O. Scherzer, **“Using the Complex Ginzburg Landau Equation for Digital Inpainting in 2D and 3D”**, Scale Space Methods in ComputerVision, LNCS 2695, pages 225–236, 2003.

43. L. Landau, V. Ginzburg, **"On the Theory of Superconductivity"**, Journal of Experimental and Theoretical Physics (USSR), 1950.
44. Haralick, R. M., **"Digital Step Edges from Zero Crossing of Second Directional Derivatives"**, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI–6(1): 58–68, 1984.
45. Clark, J. J., **"Authenticating Edges Produced by Zero–Crossing Algorithms"**, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI–11(1):43–57, 1989.
46. Torre, V. and T. Poggio, **"On edge detection"**, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI–8(2): 147–163, 1986.
47. Beckers, A. L. D., **"Parameter Estimation for Nonlinear Object Size Filters in Images"**, Master's Thesis, Faculty of Applied Physics, Delft University of Technology, The Netherlands, 1986.
48. Bernsen, J., **"Dynamic Thresholding of Grey–Level Images"**, Proceedings of 8th International Conference on Pattern Recognition, Paris, France, 1251–1255, 1986.
49. Van Vliet, L. J., I. T. Young and A. L. D. Beckers, **"An Edge Detection Model based on Nonlinear Laplace Filtering"**, Pattern Recognition and Artificial Intelligence 7 – towards an integration. Amsterdam, Elsevier Science Publishers B.V. (North-Holland). 63-73, 1988.
50. Van Vliet, L. J., I. T. Young and A. L. D. Beckers, **"A Nonlinear Laplace Operator as Edge Detector in Noisy Images",** Computer Vision, Graphics, and Image Processing 45(2): 167-195, 1989.

Summary in Arabic

الملخص العربى

تطوير خوارزمية لاستعادة القطع الناقصة في الصور الرقمية باستخدام المعادلات التفاضلية الجزئية

تتناول هذه الرسالة دراسة لمشكلة استعادة أجزاء ناقصة أو تالفة في الصور الرقمية باستخدام المعادلات التفاضلية الجزئية و الطرق الرياضية التركيبية لحل مشكلة استعادة الصور الرقمية.

واستعادة الصور الرقمية تعنى إكمال الأجزاء المتآكلة أو المفقودة (مثل التشققات أو الأجزاء التي فقدت أثناء نقل البيانات لاسلكيا) في الصور، بغرض إعادة وحدة الصور.

ولخوارزميات الاستعادة الرقمية للصور العديد من التطبيقات المهمة مثل ترميم الصور والمخطوطات الأثرية، استرجاع البيانات المفقودة في عمليات الارسال اللاسلكي، ضغط ملفات الصورالرقمية، تغيير محتويات الصور، عمل خدع و مؤثرات خاصة في الأفلام السينمائية.

وتتيح الاستعادة الرقمية للصور ملأ أجزاء متعددة من الصورة، تحتوي على هياكل مختلفة و لها خلفيات غير نمطية. بالإضافة أنه لا يفترض أي شروط محددة سلفا لترميم الجزء الناقص من الصورة.

الاستعادة الرقمية للصور تستخدم نظريات رياضية معقدة تهدف لإمداد الأجزاء الناقصة بالهيكل التحتي للصورة. وتعتمد التقنيات الحديثة على استقراء النقاط المجاورة، المعادلات التفاضلية الجزئية، و نشر درجة الانحناء لباقي أجزاء الصورة.

وقد تم فى هذه الرسالة بناء وتطوير خوارزميات بطريقتين تعتمدان على الحل العددى لمعادلات تفاضلية الجزئية والتركيبات البنائية للصور
بجانب دراستنا لمشكلة استعادة الصور الرقمية اعتمادا على حل مجموعة من المعدلات التفاضلية الجزئية، فقد اهتممنا أيضا باستعادة الصورة في حالة اذا كانت المساحات المراد استرجاعها كبيرة حيث تفشل معظم الطرق المعتمدة على النقاط المجاورة.

من خلال هذا البحث قمنا بتقديم خوارزمية مُحسنة تعتمد على استخدام معلومات من جميع أجزاء الصورة عن طريق اضافتين هامتين هما معدل تغييرشدة اللون فى أجزاء الصورة و التفاضل الثاني في اتجاه معدل التغيير.
وقد أثبتت هذه الطريقة كفائتها و فعاليتها بعد أن قمنا باختبارها على العديد من الصور ذات التركيبات و التكوينات المختلفة و المعقدة.
بمقارنة هذه النتائج مع نتائج الطرق السابقة و جدنا أن الطريقة التي قدمناها تميزت بقدرتها على استرجاع التفاصيل الصغيرة و في نفس الوقت الحفاظ على الهيكل الهندسى للمساحات المحيطة بالجزء المراد استرجاعه.

كما أن الطريقة التي توصلنا اليها من خلال هذا البحث استطاعت التغلب على المشاكل التي ظهرت مع الطرق السابقة مثل تشوه الصور و ظهور تشويش غير متوقع بالصور.

وتتكون هذه الرسالة من خمس فصول:
□□□□□ : تم فيه التقديم لمحتويات الرسالة وعرض الدوافع والاهداف للقيام بهذا البحث، كما تم عرض مختصر لبقية اجزاء الرسالة.

□□□□□ : يشتمل علي عرض لاخر ما توصل اليه الباحثون فى هذا المجال وما تم تطويره من خوارزميات و تقنيات للوصول الى أفضل نتائج ممكنة.

□□□□□ **:** يحتوي عرض مفصل وتحليل للطرق و الخوارزميات المطورة والمطبقة بالرسالة طبقناها و كيف أمكنا تلافي عيوب الطرق السابقة.

□□□ **:** يتضمن عرض لنتائج الخوارزميات المطورة بالرسالة ومقارنة وتحليل النتائج من حيث جودة الصورالمستعادة و الفترة الزمنية التي يستغرقها الخوارزم العددى.

□□□□□ □□□□□□: يعرض فيه الاستنتاجات المستخلصة من هذا البحث، كما تم عرض النقاط التى يمكن من خلالها عمل اضافات أخري فى موضوع الرسالة.

جامعة عين شمس
كلية الحاسبات والمعلومات
قسم علوم الحاسب

تطوير خوارزمية لاستعادة القطع الناقصة في الصور الرقمية باستخدام المعادلات التفاضلية الجزئية

رسالة مقدمة الى قسم علوم الحاسب
كلية الحاسبات والمعلومات جامعة عين شمس
كجزء من من متطلبات الحصول على درجة الماجستير فى الحاسبات والمعلومات

من
ياسمين نادر محمد الجلالي
بكالوريوس الحاسبات والمعلومات
(علوم الحاسب)
2002
كلية الحاسبات والمعلومات
جامعة قناة السويس

تحت اشراف

ا.د / عصام حامد عطا
استاذ علوم الحاسب
كلية الحاسبات والمعلومات
جامعة عين شمس

ا.د / تيمور نظمي
استاذ علوم الحاسب
كلية الحاسبات والمعلومات
جامعة عين شمس

د. بيه سيد الدسوقى
استاذ الرياضيات المساعد
كلية التربية ببورسعيد
جامعة قناة السويس

القاهرة 2007