Image Inpainting

Marcelo Bertalmio and Guillermo Sapiro*
Electrical and Computer Engineering, University of Minnesota

Vicent Caselles and Coloma Ballester
Escola Superior Politecnica, Universitat Pompeu Fabra

Abstract

Inpainting, the technique of modifying an image in an undetectable form, is as ancient as art itself. The goals and applications of inpainting are numerous, from the restoration of damaged paintings and photographs to the removal/replacement of selected objects. In this paper, we introduce a novel algorithm for digital inpainting of still images that attempts to replicate the basic techniques used by professional restorators. After the user selects the regions to be restored, the algorithm automatically fills-in these regions with information surrounding them. The fill-in is done in such a way that isophote lines arriving at the regions' boundaries are completed inside. In contrast with previous approaches, the technique here introduced does not require the user to specify where the novel information comes from. This is automatically done (and in a fast way), thereby allowing to simultaneously fill-in numerous regions containing completely different structures and surrounding backgrounds. In addition, no limitations are imposed on the topology of the region to be inpainted. Applications of this technique include the restoration of old photographs and damaged film; removal of superimposed text like dates, subtitles, or publicity; and the removal of entire objects from the image like microphones or wires in special effects.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—; I.3.4 [Computer Graphics]: Graphics Utilities—; I.4.4 [Image Processing and Computer Vision]: Restoration—; I.4.9 [Image Processing and Computer Vision]: Applications—;

Keywords: Image restoration, inpainting, isophotes, anisotropic diffusion.

各向异性扩散

1 Introduction

The modification of images in a way that is non-detectable for an observer who does not know the original image is a practice as old as artistic creation itself. Medieval artwork started to be restored as early as the Renaissance, the motives being often as much to bring medieval pictures "up to date" as to fill in any gaps [1, 2]. This practice is called *retouching* or *inpainting*. The object of inpainting is to reconstitute the missing or damaged portions of the work, in order to make it more legible and to restore its unity [2].

The need to retouch the image in an unobtrusive way extended naturally from paintings to photography and film. The purposes remain the same: to revert deterioration (e.g., cracks in photographs or scratches and dust spots in film), or to add or remove elements (e.g., removal of stamped date and red-eye from photographs, the infamous "airbrushing" of political enemies [3]).

Digital techniques are starting to be a widespread way of performing inpainting, ranging from attempts to fully automatic detection and removal of scratches in film [4, 5], all the way to software tools that allow a sophisticated but mostly manual process [6].

In this article we introduce a novel algorithm for automatic digital inpainting, being its main motivation to replicate the basic techniques used by professional restorators. At this point, the only user interaction required by the algorithm here introduced is to mark the regions to be inpainted. Although a number of techniques exist for the semi-automatic detection of image defects (mainly in films), addressing this is out of the scope of this paper. Moreover, since the inpainting algorithm here presented can be used not just to restore damaged photographs but also to remove undesired objects and writings on the image, the regions to be inpainted must be marked by the user, since they depend on his/her subjective selection. Here we are concerned on how to "fill-in" the regions to be inpainted, once they have been selected.[1] Marked regions are automatically filled with the structure of their surrounding, in a form that will be explained later in this paper.

2 Related work and our contribution

We should first note that classical image denoising algorithms do not apply to image inpainting. In common image enhancement applications, the pixels contain both information about the real data and the noise (e.g., image plus noise for additive 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. There is then a need to develop specific techniques to address these problems.

Mainly three groups of works can be found in the literature related to digital inpainting. The first one deals with the restoration of films, the second one is related to texture synthesis, and the third one, a significantly less studied class though very influential to the work here presented, is related to disocclusion.

Kokaram et al. [5] use motion estimation and autoregressive models to interpolate losses in films from adjacent frames. The basic idea is to copy into the gap the right pixels from neighboring frames. The technique can not be applied to still images or to films where the regions to be inpainted span many frames.

*Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA, {marcelo,guille}@ece.umn.edu

[1] In order to study the robustness of the algorithm here proposed, and not to be too dependent on the marking of the regions to be inpainted, we mark them in a very rough form with any available paintbrush software. Marking these regions in the examples reported in this paper just takes a few seconds to a non-expert user.

Hirani and Totsuka [7] combine frequency and spatial domain information in order to fill a given region with a selected texture. This is a very simple technique that produces incredible good results. On the other hand, the algorithm mainly deals with texture synthesis (and not with structured background), and requires the user to select the texture to be copied into the region to be inpainted. For images where the region to be replaced covers several different structures, the user would need to go through the tremendous work of segmenting them and searching corresponding replacements throughout the picture. Although part of this search can be done automatically, this is extremely time consuming and requires the non-trivial selection of many critical parameters, e.g., [8]. Other texture synthesis algorithms, e.g., [8, 9, 10], can be used as well to re-create a pre-selected texture to fill-in a (square) region to be inpainted.

In the group of disocclusion algorithms, a pioneering work is described in [11]. The authors presented a technique for removing occlusions with the goal of image segmentation.[2] The basic idea is to connect T-junctions at the same gray-level with elastica minimizing curves. The technique was mainly developed for simple images, with only a few objects with constant gray-levels, and will not be applicable for the examples with natural images presented later in this paper. Masnou and Morel [12] recently extended these ideas, presenting a very inspiring general variational formulation for disocclusion and a particular practical algorithm (not entirely based on PDE's) implementing some of the ideas in this formulation. The algorithm performs inpainting by joining with geodesic curves the points of the isophotes (lines of equal gray values) arriving at the boundary of the region to be inpainted. As reported by the authors, the regions to be inpainted are limited to having simple topology, e.g., holes are not allowed.[3] In addition, the angle with which the level lines arrive at the boundary of the inpainted region is not (well) preserved: the algorithm uses straight lines to join equal gray value pixels. These drawbacks, which will be exemplified later in this paper, are solved by our algorithm. On the other hand, we should note that this is the closest technique to ours and has motivated in part and inspired our work.

2.1 Our contribution

Algorithms devised for film restoration are not appropriate for our application since they normally work on relatively small regions and rely on the existence of information from several frames.

On the other hand, algorithms based on texture synthesis can fill large regions, but require the user to specify what texture to put where. This is a significant limitation of these approaches, as may be seen in examples presented later in this paper, where the region to be inpainted is surrounded by hundreds of different backgrounds, some of them being structure and not texture.

The technique we propose does not require any user intervention, once the region to be inpainted has been selected. The algorithm is able to simultaneously fill regions surrounded by different backgrounds, without the user specifying "what to put where." No assumptions on the topology of the region to be inpainted, or on the simplicity of the image, are made. The algorithm is devised for inpainting in structured regions (e.g., regions crossing through boundaries), though it is not devised to reproduce large textured areas. As we will discuss later, the combination of our proposed approach with texture synthesis techniques is the subject of current research.

[2] Since the region to be inpainted can be considered as occluding objects, removing occlusions is analogous to image inpainting.

[3] This is not intrinsic to the general variational formulation they propose, only to the specific discrete implementation they perform.

3 The digital inpainting algorithm

3.1 Fundamentals

Let Ω stand for the region to be inpainted, and $\partial\Omega$ for its boundary (note once again that no assumption on the topology of Ω is made). Intuitively, the technique we propose will prolong the isophote lines arriving at $\partial\Omega$, while maintaining the angle of "arrival." We proceed drawing from $\partial\Omega$ inward in this way, while curving the prolongation lines progressively to prevent them from crossing each other.

Before presenting the detailed description of this technique, let us analyze how experts inpaint. Conservators at the Minneapolis Institute of Arts were consulted for this work and made it clear to us that inpainting is a very subjective procedure, different for each work of art and for each professional. There is no such thing as "the" way to solve the problem, but the underlying methodology is 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 area surrounding Ω is continued into the gap, contour lines are drawn via the prolongation of those arriving at $\partial\Omega$; (3.) The different regions inside Ω, as defined by the contour lines, are filled with color, matching those of $\partial\Omega$; and (4.) The small details are painted (e.g. little white spots on an otherwise uniformly blue sky): in other words, "texture" is added.

A number of lessons can immediately be learned from these basic inpainting rules used by professionals. Our algorithm simultaneously, and iteratively, performs the steps (2.) and (3.) above.[4] We progressively "shrink" the gap Ω by prolonging inward, in a smooth way, the lines arriving at the gap boundary $\partial\Omega$.

3.2 The inpainting algorithm

We need to translate the manual inpainting concepts expressed above into a mathematical and algorithmic language. We proceed to do this now, presenting the basic underlying concepts first. The implementation details are given in the next section.

Let $I_0(i,j) : [0,M] \times [0,N] \rightarrow I\!R$, with $[0,M] \times [0,N] \subset I\!N \times I\!N$, be a discrete 2D gray level image. From the description of manual inpainting techniques, an iterative algorithm seems a natural choice. The digital inpainting procedure will construct a family of images $I(i,j,n) : [0,M] \times [0,N] \times I\!N \rightarrow I\!R$ such that $I(i,j,0) = I_0(i,j)$ and $lim_{n\rightarrow\infty} I(i,j,n) = I_R(i,j)$, where $I_R(i,j)$ is the output of the algorithm (inpainted image). Any general algorithm of that form can be written as

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

where the superindex n denotes the inpainting "time" n, (i,j) are the pixel coordinates, Δt is the rate of improvement and $I_t^n(i,j)$ stands for the update of the image $I^n(i,j)$. Note that the evolution equation runs only inside Ω, the region to be inpainted.

With this equation, the image $I^{n+1}(i,j)$ is an improved version of $I^n(i,j)$, with the "improvement" given by $I_t^n(i,j)$. As n increases, we achieve a better image. We need now to design the update $I_t^n(i,j)$.

As suggested by manual inpainting techniques, we need to continue the lines arriving at the boundary $\partial\Omega$ of the region Ω to be inpainted (see point (2) in Section 3.1). In other words, we need to smoothly propagate information from outside Ω into Ω (points (2) and (3) in Section 3.1). Being $L^n(i,j)$ the information that we want to propagate, and $\overrightarrow{N}^n(i,j)$ the propagation direction, this means that we must have

[4] In the discussion section we will argue how both steps can be performed separately, and we will also discuss step (4.).

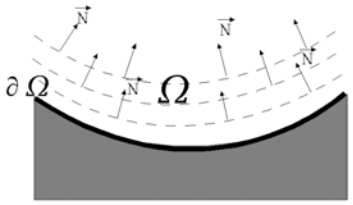

Figure 1: Propagation direction as the normal to the signed distance to the boundary of the region to be inpainted.

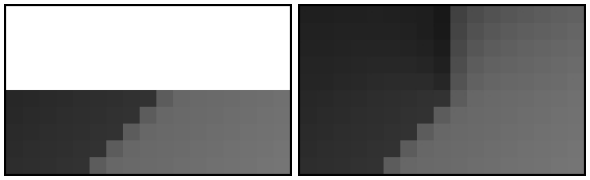

Figure 2: Unsuccessful choice of the information propagation direction. Left: detail of the original image, region to be inpainted is in white. Right: restoration.

$$I_t^n(i,j) = \overrightarrow{\delta L^n}(i,j) \cdot \overrightarrow{N}^n(i,j), \tag{2}$$

where $\overrightarrow{\delta L^n}(i,j)$ is a measure of the change in the information $L^n(i,j)$.[5] With this equation, we estimate the information $L^n(i,j)$ of our image and compute its change along the $\overrightarrow{N}$ direction. Note that at steady state, that is, when the algorithm converges, $I^{n+1}(i,j) = I^n(i,j)$ and from (1) and (2) we have that $\overrightarrow{\delta L^n}(i,j) \cdot \overrightarrow{N}^n(i,j) = 0$, meaning exactly that the information L has been propagated in the direction $\overrightarrow{N}$.

What is left now is to express the information L being propagated and the direction of propagation $\overrightarrow{N}$.

Since we want the propagation to be smooth, $L^n(i,j)$ should be an image smoothness estimator. For this purpose we may use a simple discrete implementation of the Laplacian: $L^n(i,j) := I^n_{xx}(i,j) + I^n_{yy}(i,j)$ (subscripts represent derivatives in this case). Other smoothness estimators might be used, though satisfactory results were already obtained with this very simple selection.

Then, we must compute the change $\overrightarrow{\delta L^n}(i,j)$ of this value along $\overrightarrow{N}$. In order to do this we must first define what the direction $\overrightarrow{N}$ for the 2D information propagation will be. One possibility is to define $\overrightarrow{N}$ as the normal to the signed distance to $\partial\Omega$, i.e., at each point (i,j) in Ω the vector $\overrightarrow{N}(i,j)$ will be normal to the "shrinked version" of $\partial\Omega$ to which (i,j) belongs, see Figure 1. This choice is motivated by the belief that a propagation normal to the boundary would lead to the continuity of the isophotes at the boundary. Instead, what happens is that the lines arriving at $\partial\Omega$ curve in order to align with $\overrightarrow{N}$, see Figure 2. This is of course not what we expect. Note that the orientation of $\partial\Omega$ is not intrinsic to the image geometry, since the region to be inpainted is arbitrary.

If isophotes tend to align with $\overrightarrow{N}$, the best choice for $\overrightarrow{N}$ is then the isophotes directions. This is a bootstrapping problem: having the isophotes directions inside Ω is equivalent to having the inpainted image itself, since we can easily recover the gray level image from its isophote direction field (see the discussion section and [13]).

[5] Borrowing notation from continuous mathematics, we could also write $\overrightarrow{\delta L^n}(i,j)$ as ∇L.

We use then a time varying estimation of the isophotes direction field: for any given point (i,j), the discretized gradient vector $\nabla I^n(i,j)$ gives the direction of largest spatial change, while its 90 degrees rotation $\nabla^\perp I^n(i,j)$ is the direction of smallest spatial change, so the vector $\nabla^\perp I^n(i,j)$ gives the isophotes direction. Our field $\overrightarrow{N}$ is then given by the time-varying $\overrightarrow{N}(i,j,n) = \nabla^\perp I^n(i,j)$. We are using a time-varying estimation that is coarse at the beginning but progressively achieves the desired continuity at $\partial\Omega$, instead of a fixed field $\overrightarrow{N}(i,j)$ that would imply to know the directions of the isophotes from the start.

Note that the direction field is not normalized, its norm is the norm of the gradient of $I^n(i,j)$. This choice helps in the numerical stability of the algorithm, and will be discussed in the following subsection.

Since we are performing inpainting along the isophotes, it is irrelevant if $\nabla^\perp I^n(i,j)$ is obtained as a clockwise or counterclockwise rotation of $\nabla I^n(i,j)$. In both cases, the change of $I^n(i,j)$ along those directions should be minimum.

Recapping, we estimate a variation of the smoothness, given by a discretization of the 2D Laplacian in our case, and project this variation into the isophotes direction. This projection is used to update the value of the image inside the region to be inpainted.

To ensure a correct evolution of the direction field, a diffusion process is interleaved with the image inpainting process described above.[6] That is, every few steps (see below), we apply a few iterations of image diffusion. This diffusion corresponds to the periodical curving of lines to avoid them from crossing each other, as was mentioned in Section 3.1. We use anisotropic diffusion, [14, 15], in order to achieve this goal without losing sharpness in the reconstruction. In particular, we apply a straightforward discretization of the following continuous-time/continuous-space anisotropic diffusion equation:

$$\frac{\partial I}{\partial t}(x,y,t) = g_\epsilon(x,y)\kappa(x,y,t)\,|\nabla I(x,y,t)|\,, \forall (x,y) \in \Omega^\epsilon \tag{3}$$

where Ω^ϵ is a dilation of Ω with a ball of radius ϵ, κ is the Euclidean curvature of the isophotes of I and $g_\epsilon(x,y)$ is a smooth function in Ω^ϵ such that $g_\epsilon(x,y) = 0$ in $\partial\Omega^\epsilon$, and $g_\epsilon(x,y) = 1$ in Ω (this is a way to impose Dirichlet boundary conditions for the equation (3)).[7]

3.3 Discrete scheme and implementation details

The only input to our algorithm are the image to be restored and the mask that delimits the portion to be inpainted. As a preprocessing step, the *whole* original image undergoes anisotropic diffusion smoothing. The purpose of this is to minimize the influence of noise on the estimation of the direction of the isophotes arriving at $\partial\Omega$. After this, the image enters the inpainting loop, where only the values inside Ω are modified. These values change according to the discrete implementation of the inpainting procedure, which we proceed to describe. Every few iterations, a step of anisotropic diffusion is applied (a straightforward, central differences implementation of (3) is used; for details see [14, 15]). This process is repeated until a steady state is achieved.

Let $I^n(i,j)$ stand for each one of the image pixels inside the region Ω at the inpainting "time" n. Then, the discrete inpainting equation borrows from the numerical analysis literature and is given by

$$I^{n+1}(i,j) = I^n(i,j) + \Delta t I_t^n(i,j), \forall (i,j) \in \Omega \tag{4}$$

[6] We can also add the diffusion as an additional term in $I_t^n(i,j)$, the results being very similar.

[7] Other filters, e.g., form mathematical morphology, can be applied as well, though we found the results obtained with this equation satisfactory.

where

$$I_t^n(i,j) = \left(\overrightarrow{\delta L^n}(i,j) \cdot \frac{\overrightarrow{N}(i,j,n)}{\left|\overrightarrow{N}(i,j,n)\right|} \right) |\nabla I^n(i,j)|\,, \qquad (5)$$

$$\overrightarrow{\delta L^n}(i,j) := (L^n(i{+}1,j){-}L^n(i{-}1,j), L^n(i,j{+}1){-}L^n(i,j{-}1)), \qquad (6)$$

$$L^n(i,j) = I^n_{xx}(i,j) + I^n_{yy}(i,j), \qquad (7)$$

$$\frac{\overrightarrow{N}(i,j,n)}{\left|\overrightarrow{N}(i,j,n)\right|} := \frac{(-I^n_y(i,j), I^n_x(i,j))}{\sqrt{(I^n_x(i,j))^2 + (I^n_y(i,j))^2}}, \qquad (8)$$

$$\beta^n(i,j) = \overrightarrow{\delta L^n}(i,j) \cdot \frac{\overrightarrow{N}(i,j,n)}{\left|\overrightarrow{N}(i,j,n)\right|}, \qquad (9)$$

and

$$|\nabla I^n(i,j)| = \begin{cases} \sqrt{(I^n_{xbm})^2 + (I^n_{xfM})^2 + (I^n_{ybm})^2 + (I^n_{yfM})^2}, \\ \text{when } \beta^n > 0 \\ \sqrt{(I^n_{xbM})^2 + (I^n_{xfm})^2 + (I^n_{ybM})^2 + (I^n_{yfm})^2}, \\ \text{when } \beta^n < 0 \end{cases} \qquad (10)$$

We first compute the 2D smoothness estimation L in (7) and the isophote direction $\overrightarrow{N}/\left|\overrightarrow{N}\right|$ in (8). Then in (9) we compute β^n, the projection of $\overrightarrow{\delta L}$ onto the (normalized) vector $\overrightarrow{N}$, that is, we compute the change of L along the direction of $\overrightarrow{N}$. Finally, we multiply β^n by a *slope-limited* version of the norm of the gradient of the image, $|\nabla I|$, in (10). [8] A central differences realization would turn the scheme unstable, and that is the reason for using slope-limiters. The subindexes b and f denote backward and forward differences respectively, while the subindexes m and M denote the minimum or maximum, respectively, between the derivative and zero (we have omitted the space coordinates (i,j) for simplicity); see [16] for details. Finally, let us note that the choice of a non-normalized field $\overrightarrow{N}$ instead of a normalized version of it allows for a simpler and more stable numerical scheme; see [17, 18].

Note once again that when the inpainting algorithm arrives to steady state, that is, $I_t = 0$, we have geometrically solved $\nabla(\text{Smoothness}) \cdot \nabla^\perp I = 0$, meaning that the "smoothness" is constant along the isophotes. [9]

When applying equations (4)-(10) to the pixels in the border $\partial\Omega$ of the region Ω to be inpainted, known pixels from outside this region are used. That is, conceptually, we compute equations (4)-(10) in the region Ω^ϵ (an ϵ dilation of Ω), although we update the values only inside Ω (that is, (4) is applied only inside Ω). The information in the narrow band $\Omega^\epsilon - \Omega$ is propagated inside Ω. Propagation of this information, both gray-values and isophotes directions, is fundamental for the success of the algorithm.

In the restoration loop we perform A steps of inpainting with (4), then B steps of diffusion with (3), again A steps of (4), and so on. The total number of steps is T. This number may be preestablished, or the algorithm may stop when changes in the image are below a given threshold. The values we use are: $A = 15$, $B = 2$, at speed $\Delta t = 0.1$. The value of T depends on the size of Ω.

[8] Note that $|\nabla I| = |\nabla^\perp I|$.

[9] This type of information propagation is related and might be applicable to velocity fields extension in level-set techniques [19, 20].

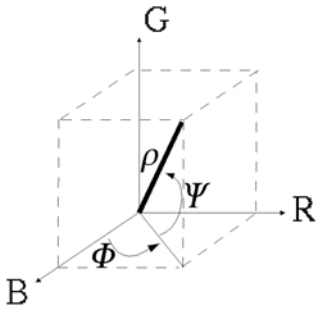

Figure 3: Relation between the (R,G,B) color model and the one used in this article,$(\rho, sin\phi, sin\psi)$.

If Ω is of considerable size, a *multiresolution* approach is used to speed-up the process.[10]

Color images are considered as a set of three images, and the above described technique is applied independently to each one. To avoid the appearance of spurious colors, we use a color model which is very similar to the LUV model, with one luminance and two chroma components. See Figure 3.

4 Results

The CPU time required for inpainting depends on the size of Ω. In all the color examples here presented, the inpainting process was completed in less than 5 minutes (for the three color planes), using non-optimized C++ code running on a PentiumII PC (128Mb RAM, 300MHz) under Linux. All the examples use images available from public databases over the Internet. The main examples here presented, and additional ones, can be seen at http://www.ece.umn.edu/users/marcelo/restoration.html, where in addition to the original and inpainted images reproduced below, the evolution process can be observed.

Figure 4 shows, on the left, a synthetic image with the region to inpaint in white. Here Ω is large (30 pixels in diameter) and contains a hole. The inpainted reconstruction is shown on the right. Notice that contours are recovered, joining points from the inner and outer boundaries. Also, these reconstructed contours follow smoothly the direction of the isophotes arriving at $\partial\Omega$ (the algorithm reported in [12] will fail with this type of data).

Figure 5 shows a deteriorated B&W image (first row) and its reconstruction (second row). As in all the examples in this article, the user only supplied the "mask" image (last row). This mask was drawn manually, using a paintbrush-like program. The variables were set to the values specified in the previous section, and the number of iterations T was set to 3000. When multiresolution is not used, the CPU time required by the inpainting procedure was approximately 7 minutes. With a 2-level multiresolution scheme, only 2 minutes were needed. Observe that details in the nose and right eye of the middle girl could not be completely restored. This is in part due to the fact that the mask covers most of the relevant information, and there is not much to be done without the use of high level prior information (e.g., the fact that it is an eye). These minor errors can be corrected by the manual procedures mentioned in the introduction, and still the overall inpainting time would be reduced by orders of magnitude. This example was tested and showed to be robust to initial conditions inside the region to be inpainted.

Figure 6 shows a vandalized image and its restoration, followed by an example where overimposed text is removed from the image.

[10] We basically use the converged result of a lower resolution stage to initialize the higher one, as classically done in image processing.

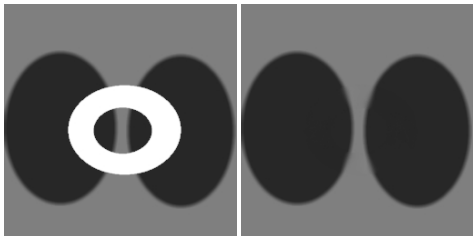

Figure 4: Synthetic example: Ω is shown in white. Topology is not an issue, and the recovered contours smoothly continue the isophotes.

These are typical examples where texture synthesis algorithms as those described in the introduction can not be used, since the number of different regions to be filled-in is very large.

The next figure shows the progressive nature of the algorithm, several intermediate steps of the inpainting procedure are shown, removing painted text over a natural scene.

Finally, Figure 8 shows an entertainment application. The bungee cord and the knot tying the man's legs have been removed. Given the size of Ω a 2-level multiresolution scheme was used. Here it becomes apparent that it is the user who has to supply the algorithm with the masking image, since the choice of the region to inpaint is completely subjective.

5 Conclusions and future work

In this paper we have introduced a novel algorithm for image inpainting that attempts to replicate the basic techniques used by professional restorators. The basic idea is to smoothly propagate information from the surrounding areas in the isophotes direction. The user needs only to provide the region to be inpainted, the rest is automatically performed by the algorithm in a few minutes. The inpainted images are sharp and without color artifacts. The examples shown suggest a wide range of applications like restoration of old photographs and damaged film, removal of superimposed text, and removal of objects. The results can either be adopted as a final restoration or be used to provide an initial point for manual restoration, thereby reducing the total restoration time by orders of magnitude.

One of the main problems with our technique is the reproduction of large textured regions, as can be seen in Figure 9. The algorithm here proposed is currently being tested in conjunction with texture synthesis ideas to address this issue. We are mainly investigation the combination of this approach with the reaction-diffusion ideas of Kass and Witkin and of Turk. An ideal algorithm should be able to automatically switch between textured and geometric areas, and select the best suited technique for each region.

We would also like to investigate how to inpaint from partial degradation. In the example of the old photo for example, ideally the mask should not be binary, since some underlying information exists in the degraded areas.

The inpainting algorithm here presented has been clearly motivated by and has borrowed from the intensive work on the use of Partial Differential Equations (PDE's) in image processing and computer vision. When "blindly" letting the grid go to zero, the inpainting technique in equations (4)-(10) naively resembles a third order equation, for which too many boundary conditions are imposed (being all of them essential). Although theoretical results for high order equations are available, e.g., [21], and some properties like preservation of the image moments can be immediately proved for our corresponding equation (this was done by A. Bertozzi), further formal study of our "high order equation" is needed (see also [22, 23]). Nevertheless, this suggests the investigation of the use of lower, second order, PDE's to address the inpainting problem. We can split the inpainting problem into two coupled variational formulations, one for the isophotes direction (point (2) in Section 3.1) and one for the gray-values, consistent with the estimated directions (point (3) in Section 3.1). The corresponding gradient descent flows will give two coupled second order PDE's for which formal results regarding existence and uniqueness of the solutions can be shown. This is reported in [24].

Acknowledgments

This work started when the authors were visiting the Institute Henri Poincare in Paris, France. We thank the organizers of the quarter on "Mathematical Questions in Signal and Image Processing" and the Institute for their hospitality and partial financial support. We would like to thank Tom Robbins and Elizabeth Buschor from the Upper Midwest Conservation Association for their help in linking our work with actual art restoration and conservation. Amy Miller, of Photo-Medic, kindly provided the damaged photograph shown in the examples. We also thank Dr. Santiago Betelu, Prof. Stan Osher, Prof. Eero Simoncelli, and Prof. Andrea Bertozzi for very interesting discussions and feedback. The reviewers provided many useful ideas. This work was partially supported by a grant from the Office of Naval Research ONR-N00014-97-1-0509, the Office of Naval Research Young Investigator Award, the Presidential Early Career Awards for Scientists and Engineers (PECASE), a National Science Foundation CAREER Award, by the National Science Foundation Learning and Intelligent Systems Program (LIS), and Universidad de la Republica (Uruguay).

References

[1] S. Walden. *The Ravished Image*. St. Martin's Press, New York, 1985.

[2] G. Emile-Male. *The Restorer's Handbook of Easel Painting*. Van Nostrand Reinhold, New York, 1976.

[3] D. King. *The Commissar Vanishes*. Henry Holt and Company, 1997.

[4] A.C. Kokaram, R.D. Morris, W.J. Fitzgerald, P.J.W. Rayner. *Detection of missing data in image sequences*. IEEE Transactions on Image Processing 11(4), 1496-1508, 1995.

[5] A.C. Kokaram, R.D. Morris, W.J. Fitzgerald, P.J.W. Rayner. *Interpolation of missing data in image sequences*. IEEE Transactions on Image Processing 11(4), 1509-1519, 1995.

[6] C. Braverman. *Photoshop retouching handbook*. IDG Books Worldwide, 1998.

[7] A. Hirani and T. Totsuka. *Combining Frequency and spatial domain information for fast interactive image noise removal*. Computer Graphics, pp. 269-276, SIGGRAPH 96, 1996.

[8] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," *Proc. IEEE International Conference Computer Vision*, pp. 1033-1038, Corfu, Greece, September 1999.

[9] D. Heeger and J. Bergen. *Pyramid based texture analysis/synthesis*. Computer Graphics, pp. 229-238, SIGGRAPH 95, 1995.

[10] E. Simoncelli and J. Portilla. *Texture characterization via joint statistics of wavelet coefficient magnitudes*. 5th IEEE Int'l Conf. on Image Processing, Chicago, IL. Oct 4-7, 1998.

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

[12] S. Masnou and J.M. Morel. *Level-lines based disocclusion*. 5th IEEE Int'l Conf. on Image Processing, Chicago, IL. Oct 4-7, 1998.

[13] C. Kenney and J. Langan. *A new image processing primitive: reconstructing images from modified flow fields*. University of California Santa Barbara Preprint, 1999.

[14] P. Perona and J. Malik *Scale-space and edge detection using anisotropic diffusion*. IEEE-PAMI 12, pp. 629-639, 1990.

[15] L. Alvarez, P.L. Lions, J.M. Morel. *Image selective smoothing and edge detection by nonlinear diffusion*. SIAM J. Numer. Anal. 29, pp. 845-866, 1992.

[16] S. Osher and J. Sethian. *Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations*. Journal of Computational Physics, 79:12-49, 1988.

[17] A. Marquina and S. Osher. *Explicit algorithms for a new time dependent model based on level set motion for nonlinear debluring and noise removal*. UCLA CAM Report 99-5, January 1999.

[18] L. Rudin, S. Osher and E. Fatemi. *Nonlinear total variation based noise removal algorithms*. Physica D, 60, pp. 259-268, 1992.

[19] S. Osher, personal communication, October 1999.

[20] H. K. Zhao, T. Chan, B. Merriman, and S. Osher, "A variational level-set approach to multiphase motion," *J. of Computational Physics* **127**, pp. 179-195, 1996.

[21] A. Bertozzi *The mathematics of moving contact lines in thin liquid films*. Notices Amer. Math. Soc., Volume 45, Number 6, pp. 689-697, June/July 1998.

[22] J. Tumblin and G. Turk, "LCIS: A boundary hierarchy for detail-preserving contrast reduction," Computer Graphics, pp. 83-90, SIGGRAPH 99, 1999.

[23] T. Chan and J. Shen, "Mathematical models for local deterministic inpaintings," *UCLA CAM TR* **00-11**, March 2000.

[24] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and grey levels," *University of Minnesota IMA TR*, April 2000.

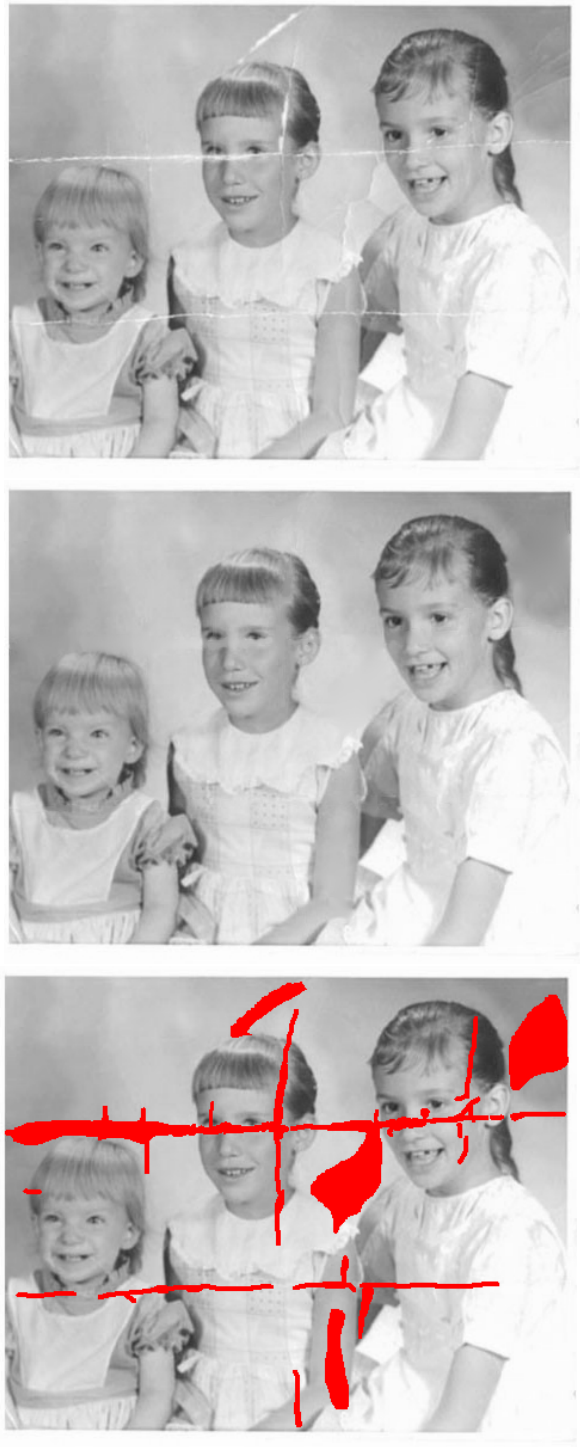

Figure 5: Restoration of an old photograph.

Figure 6: Restoration of a color image and removal of superimposed text.

Figure 7: Progressive nature of the algorithm. Several intermediate steps of the reconstruction are shown.

Figure 8: The bungee cord and the knot tying the man’s feet have been removed.

Figure 9: Limitations of the algorithm: texture is not reproduced.