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.

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 inpainting, ranging from attempts to fully automatic detection and removal of scratches in film [4, 5, 6], all the way to software tools that allow a sophisticated but mostly manual process [7].

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, since the regions to be inpainted are usually large. That is, regions occupied by top to bottom scratches along several film frames, long cracks in photographs, superimposed large fonts, and so on, are of significant larger size than the type of noise assumed in common image enhancement algorithms. In addition, 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.

Joyeux et al. [4] devised a 2-steps frequency based reconstruction system for detecting and removing line scratches in films. They propose to first recover low and then high frequencies. Although good results are obtained for their specific application, the algorithm can not handle large loss areas. Frequency domain algorithms trade a good recovery of the overall structure of the regions for poor spatial results regarding, for instance, the continuity of lines.

Kokaram et al. [6] 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.

Hirani and Totsuka [8] 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

*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.

(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., [9]. Other texture synthesis algorithms, e.g., [9, 10, 11], can be used as well to re-create a preselected texture to fill-in a (square) region to be inpainted.

In the group of disocclusion algorithms, a pioneering work is described in [12]. 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 [13] recently extended these ideas, presenting a very inspiring general variational formulation for disocclusion and a particular practical algorithm implementing some of the ideas in this formulation. The algorithm performs inpainting by joining with geodesic curves the points of the isophotes 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 will be exemplified later in this paper. 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.

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).

[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.

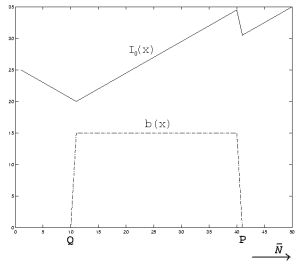

Figure 1: 1D signals $I_0(i)$ and $b(i)$.

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.[4]

Before presenting the detailed description of this technique, let us analyze how experts inpaint. Conservators of 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.

Our algorithm simultaneously and iteratively performs the steps (2.) and (3.) above. (In the discussion section we will argue how both steps can be performed separately, and we will also discuss step (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 A one dimensional illustration

A simple one dimensional (1D) example is first used to illustrate our technique. This is done only for a didactic purpose, since our algorithm was devised for 2D, and there are other techniques (such as splines) that might yield better interpolation results in 1D. Our 1D example will help to explain how we translate the basic inpainting ideas presented above into a working algorithm. Consider a 1D (discrete) signal $I_0(i) : [0, M] \rightarrow I\!R$ ($[0, M] \subset I\!N$) and a "box-function" mask signal $b(i)$, as illustrated in Figure 1. The region Ω to inpaint is given by the set of points where $b(i) \neq 0$, while its boundary $\partial\Omega$ are the points Q and P. Let $\overrightarrow{N}$ be the direction of increasing i.

The progressive nature of the inpainting procedure suggests an "evolution" approach. We define a family of 1D signals $I(i, n) : [0, M] \times [0, \infty) \rightarrow I\!R$ ($[0, M] \subset I\!N, [0, \infty] \subset I\!N$) where n is the "time" parameter, such that $I(i, 0) = I_0(i)$ and $lim_{n\rightarrow\infty} I(i, n) = I_R(i)$, where $I_R(i)$ is the output of the algorithm (inpainted image). At each iteration step n we want to construct $I(i, n)$ given $I(i, n - 1)$.

For the 1D case, the "prolongation of lines into Ω" is interpreted as "the continuation of the function $I_0(i)$ from the left of Q to its right, and from the right of P to its left, in a smooth way." Therefore,

[4] This type of information propagation is related and might be applicable to velocity fields extension in level-set techniques [14].

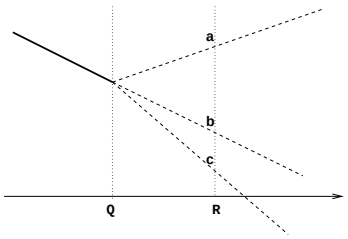

Figure 2: Discrete example; see text..

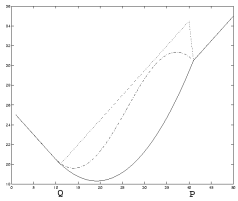

Figure 3: Three stages of the evolution of $I(i,n)$ in Figure 1.

in order to inpaint in the direction of $\overrightarrow{N}$, the following pseudo-code realizes the ideas above:

for $n = 0$ to $n = Number - of - Iterations$
  for each point (pixel) $(i) \in \Omega$
   compute the smoothness of $I^n(i)$: $L^n(i) := I^n_{xx}(i)$
   compute the smoothness of $I^n(i-1)$: $L^n(i-1) := I^n_{xx}(i-1)$
   compute the smoothness variation in $-\overrightarrow{N}$:
        $I^n_t(i) := L^n(i) - L^n(i-1)$
   $I^{n+1}(i) = I^n(i) + \Delta t I^n_t(i)$
  end
end,

where the index i denotes spatial position, the superindex n denotes the artificial evolution time ($I(i,n) = I^n(i)$), the subindex x denotes (discretized) spatial differentiation in the horizontal axis direction, and Δt controls the velocity of the evolution. We are changing in time the function $I(i,n)$ so that at each point i, $I^{n+1}(i)$ tends to follow $I^n(i-1)$. The purpose of the smoothness estimation is to ensure continuity not only of $I(i,n)$ but also of its derivatives. Our discrete smoothness estimator is the second spatial derivative of $I(i,n)$, $L^n(i) = I^n_{xx}(i)$, which in discrete form can be written as $L^n(i) = I^n(i-1) - 2I^n(i) + I^n(i+1)$.

To explain why this works, consider Figure 2. The continuous line denotes the part of $I(i,n)$ that is not changed by the algorithm since it is outside Ω. To the right of Q, the three dashed lines denote three possible initial conditions for $I(x,0)$ inside Ω. Being straight lines, $L(R,0) = 0$ in the three cases. At the point Q on the other hand, the value of $L(Q,0)$ is different in each case:

If $I(R,0) = a$, $L(Q,0) > 0$, $I_t(R,0) = L(R,0) - L(Q,0) < 0$, and $I(R,0)$ decreases;
If $I(R,0) = b$, $L(Q,0) = 0$, $I_t(R,0) = L(R,0) - L(Q,0) = 0$, and $I(R,0)$ does not change;
If $I(R,0) = c$, $L(Q,0) < 0$, $I_t(R,0) = L(R,0) - L(Q,0) > 0$, and $I(R,0)$ increases.

In all three cases the evolution is deforming $I(i,n)$ inside Ω into a continuation of $I_0(i)$ to the left of Q.

Figure 3 shows three stages of the evolution of the signal in Figure 1. The dotted line corresponds to the initial condition, $I(i,0)$. The dashed line corresponds to an intermediate state. The continuous line corresponds to $I_R(i)$, the steady state of the evolution. Notice how continuity up to the first derivative is obtained for the point Q. That is not the case for P, since the direction of propagation is $\overrightarrow{N}$ and not $-\overrightarrow{N}$. We will later see that this is not a problem for the 2D case. We have tested 1D examples with different initial conditions, and found that the algorithm always converges to the same solution within a 0.1% error margin. The formal study of this experimental result is of theoretical interest (see Section 5).

3.3 The 2D inpainting algorithm

Let us re-write the 1D discrete equation describing the rate of change of $I(i,n)$:

$$I^{n+1}(i) = I^n(i) + \Delta t I^n_t(i), \tag{1}$$

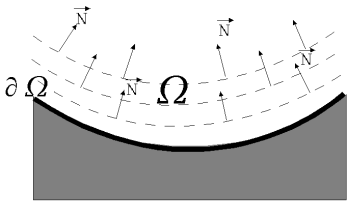

Figure 4: Propagation direction as the normal to the signed distance to the boundary of the region to be inpainted.

where

$$I^n_t(i) = L^n(i) - L^n(i-1), \ \ \forall i \in \Omega, \tag{2}$$

$$L^n(i) = I^n(i-1) - 2I^n(i) + I^n(i+1). \tag{3}$$

In other words, we estimate the smoothness $L^n(i)$ of our function and compute its change along the $-\overrightarrow{N}$ direction, adapting the image in the region to be inpainted with this variation of smoothness quantity. Let us now see how to extend these ideas into 2D.

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. The inpainting procedure will construct a family of images $I(i,j,n) : [0,M] \times [0,N] \times [0,\infty) \rightarrow I\!R$ 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).

Following the 1D procedure, first we estimate the smoothness $L^n(i,j)$ of our function. For this purpose we may use a simple discrete implementation of the Laplacian: $L^n(i,j) := I^n_{xx}(i,j) + I^n_{yy}(i,j)$. Other smoothness estimators might be used, though satisfactory results were already obtained with this very simple selection.

Then, we must compute the change 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 4. 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 5. 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 [15]).

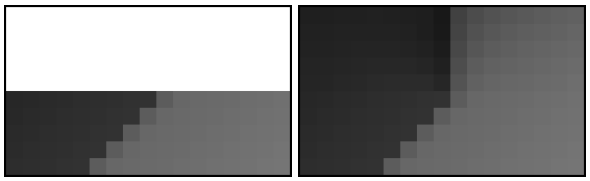

Figure 5: Unsuccessful choice of the information propagation direction. Left: detail of the original image, region to be inpainted is in white. Right: restoration.

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 field is not normalized as it was in the 1D case, where $\overrightarrow{N}$ was just the direction of increasing i. The reason for this choice relies on the numerical stability of the algorithm, and will be discussed in the following subsection.

Since we are performing an 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. 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, [16, 17], 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:

$$\frac{\partial I}{\partial t}(x, y, t) = g_\epsilon(x, y)\kappa(x, y, t)\left|\nabla I(x, y, t)\right|, \forall (x, y) \in \Omega^\epsilon \quad (4)$$

where Ω^ϵ is a dilation of Ω with a ball of radius ϵ, κ is the Euclidean curvature of the isophotes of u and $g_\epsilon(x, y)$ is a smooth function in Ω^ϵ such that $g_\epsilon(x, y) = 0$ in $\Omega^\epsilon \setminus \Omega$, and $g_\epsilon(x, y) = 1$ at the set of points of Ω whose distance to $\partial\Omega$ is larger that ϵ (this is a way to impose Dirichlet boundary conditions for the equation (4)).

3.4 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 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 (4) is used; for details see [16, 17]). 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

$$I^{n+1}(i, j) = I^n(i, j) + \Delta t I_t^n(i, j), \quad (5)$$

where

$$I_t^n(i, j) = \left(\overrightarrow{\delta L^n}(i, j) \cdot \frac{\overrightarrow{N}(i, j, n)}{\left|\overrightarrow{N}(i, j, n)\right|}\right)\left|\nabla I^n(i, j)\right|, \quad (6)$$

$$\overrightarrow{\delta L^n}(i, j) := (L^n(i{+}1, j){-}L^n(i{-}1, j), L^n(i, j{+}1){-}L^n(i, j{-}1)), \quad (7)$$

$$L^n(i, j) = I_{xx}^n(i, j) + I_{yy}^n(i, j), \quad (8)$$

$$\frac{\overrightarrow{N}(i, j, n)}{\left|\overrightarrow{N}(i, j, n)\right|} := \frac{(-I_y^n(i, j), I_x^n(i, j))}{\sqrt{(I_x^n(i, j))^2 + (I_y^n(i, j))^2 + \epsilon}}, \quad (9)$$

$$\beta^n(i, j) = \overrightarrow{\delta L^n}(i, j) \cdot \frac{\overrightarrow{N}(i, j, n)}{\left|\overrightarrow{N}(i, j, n)\right|}, \quad (10)$$

and

$$\left|\nabla I^n(i, j)\right| = \begin{cases} \sqrt{(I_{xbm}^n)^2 + (I_{xfM}^n)^2 + (I_{ybm}^n)^2 + (I_{yfM}^n)^2}, \\ \text{when } \beta^n > 0 \\ \sqrt{(I_{xbM}^n)^2 + (I_{xfm}^n)^2 + (I_{ybM}^n)^2 + (I_{yfm}^n)^2}, \\ \text{when } \beta^n < 0 \end{cases} \quad (11)$$

We first compute the 2D smoothness estimation L in (8) and the isophote direction $\overrightarrow{N}/\left|\overrightarrow{N}\right|$ in (9). Then in (10) 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 (11). 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 [18] for details. Finally, let us note that the choice of $\nabla^\perp I$ instead of a normalized version of it (as was the case in 1D) allows for a simpler and more stable numerical scheme; see [19, 20].

When applying equations (5)-(11) 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 (5)-(11) in the region Ω^ϵ (an ϵ dilation of Ω), although update the values only inside Ω (that is, (5) is applied only inside Ω). The information in the narrow band $\Omega^\epsilon - \Omega$ (isophote directions and gray values) 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 (5), then B steps of diffusion with (4), again A steps of (5), and so on. The total number of steps is T. This number may be pre-established, or the algorithm may stop when changes in the image

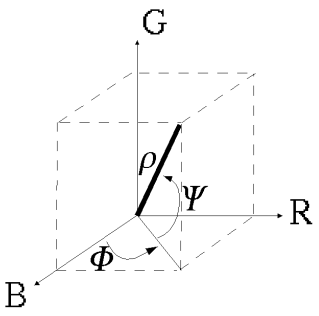

Figure 6: Relation between the (R, G, B) color model and the one used in this article,$(\rho, sin\phi, sin\psi)$.

are below a given threshold. The values we use are: $A = 15, B = 2, \Delta t = 0.1$. The value of T depends on the size of Ω. If Ω is of considerable size, a *multiresolution* approach is used to speed-up the process.

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 6.

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.

Figure 7 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 $\partial\Omega$.

Figure 8 shows a deteriorated B&W image and its reconstruction. As in all the examples in this article, the user only supplied the "mask" image, shown in Figure 9. This image 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. In Figure 10, top, we show a different initial condition inside Ω. The inpainted image (bottom) is practically identical to the one in Figure 9, thereby showing the robustness of the algorithm.

Figure 11 shows a vandalized image and its restoration, followed by an example where overimposed text is removed from the image. 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 is very large.

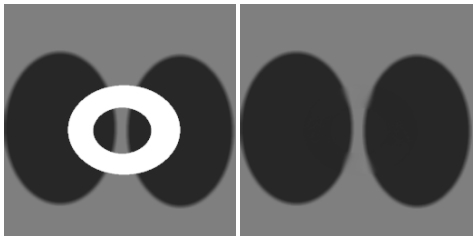

Figure 7: Synthetic example: Ω is shown in white. Topology is not an issue, and the recovered contours smoothly continue the isophotes.

Figure 12 shows the progressive nature of the algorithm: several intermediate steps of the inpainting procedure are shown for a detail of figure 11.

Finally, Figure 13 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 14. The algorithm here proposed is currently being tested in conjunction with texture synthesis ideas to address this issue.

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 (5)-(11), naively resembles a third order equation (not resulting from a variational gradient descent flow, see below). Although adding regularization terms might make this high order equation stable and allow a formal analysis,[5] to the best of our knowledge the complete understanding of such type of equations is not just beyond the scope of this paper but beyond the current state of mathematical knowledge (although results for other high order equations, which might be relevant for image processing as well, are available, e.g., [22]). In addition, the discrete algorithm here introduced uses as "boundary conditions" both gray values and isophote directions present at the narrow band surrounding the region to be inpainted. This is done, as explained above, performing the computations in the dilated region Ω^ϵ. All this important information needs to be incorporated to such an equation as "extra" (and

[5] Note that numerical implementations of PDE's intrinsically regularize the equations.

fundamental) boundary conditions. Nevertheless, this suggests the investigation of the use of lower, second order, PDE's to address the inpainting problem. Moreover, having these equations to be gradient descent flows of variational formulations permits the inclusion of the narrow band information simply by changing the limits of integration. One possible approach to follow is to first reconstruct the isophote directions θ (a planar vector field), and from them the corresponding gray values I. Once the isophote directions are reconstructed, the gradient direction is given, and we can reconstruct the gray levels by finding an image consistent with these gradient directions. Although this can be done using a one dimensional transport equation, the existent theory limits the possible gradient fields, thereby disqualifying edges. We can take then a variational approach and find the image I minimizing $\int_{\Omega^\epsilon}(|\nabla I| - \theta \cdot \nabla I)d\Omega$, where θ is the reconstructed gradient direction inside Ω^ϵ (an additional term that penalizes the deviation from the real image in the narrow band $\Omega^\epsilon - \Omega$ can be added as well). The minimal I is obtained via gradient descent flows (second order PDE). Therefore, what is left is to reconstruct the vector field θ. This can again be achieved using a variational formulation on θ with the proper boundary conditions to guarantee the continuity of the vector field across $\partial\Omega$ (see also [13, 21]). Both variational formulations can actually be coupled to simultaneously recover the isophotes (vector field) and gray values via the corresponding gradient descent flow, obtaining a set of coupled second order PDE's.[6] Preliminary results in this direction are promising and will be further investigated and reported elsewhere [23]. One of the advantages of this approach is that formal results regarding uniqueness and existence of the solutions can be shown, thereby reducing the necessity to rely mainly on experimental validations as done until now with the basic image inpainting algorithms reported in the literature.

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. We also thank Dr. Santiago Betelu, University of Minnesota, Prof. Stan Osher, UCLA, and Prof. Eero Simoncelli, NYU, for very interesting discussions and feedback. 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, and by the National Science Foundation Learning and Intelligent Systems Program (LIS).

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.

[6] Intuitively, it is clear that we can not address the inpainting problem with a single second order PDE, since regularization constraints on vector fields are needed (in [13], the general variational formulation suggested will normally lead to a fourth order gradient descent flow).

[4] L. Joyeux, O. Buisson, B. Besserer, S. Boukir. *Detection and Removal of Line Scratches in motion Picture Films*. Proceedings of CVPR'99, IEEE Int. Conf. on Computer Vision and Pattern Recognition, Fort Collins, Colorado, USA, June 1999.

[5] 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.

[6] 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.

[7] C. Braverman. *Photoshop retouching handbook.* IDG Books Worlwide, 1998.

[8] 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.

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

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

[11] 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.

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

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

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

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

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

[17] 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.

[18] 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.

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

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

[21] L. I. Rudin and S. Osher, *Total variation based image restoration with free local constraints.* Proc. IEEE-ICIP I, pp. 31-35, Austin, Texas, 1994.

[22] 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.

[23] C. Ballester, M. Bertalmio, V. Caselles, and G. Sapiro, “Interpolation of planar vector fields and applications,” in preparation.

Figure 8: Restoration of an old photograph.

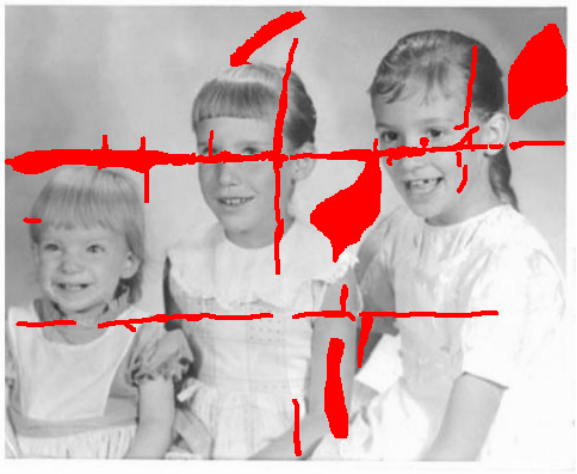

Figure 9: The user defines the region to inpaint (here shown in red).

Figure 10: From a different initial condition inside Ω we arrive at a very similar result..

Figure 11: Restoration of a color image and removal of superimposed text.

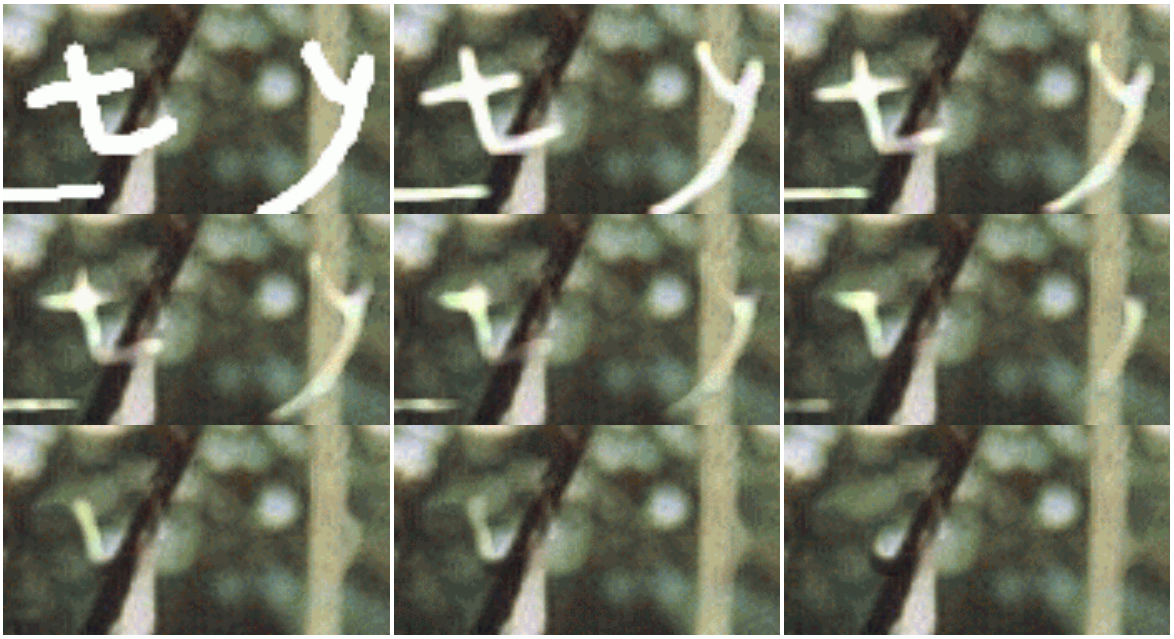

Figure 12: Progressive nature of the algorithm: several intermediate steps of the reconstruction of figure 11 (detail).

Figure 13: The bungee cord and the knot tying the man's feet have been removed.

Figure 14: Limitations of the algorithm: texture is not reproduced.