Image inpainting using a TV-Stokes equation*

Xue-Cheng Tai†, Stanley Osher‡ and Randi Holm§

January 3, 2006

Abstract

Based on some geometrical considerations, we propose a two-step method to do digital image inpainting. In the first step, we try to propagate the isophote directions into the inpainting domain. An energy minimization model combined with the zero divergence condition is used to get a non-linear Stokes equation. Once the isophote directions are constructed, an image is restored to fit the constructed directions. Both steps reduce to the solving of some nonlinear partial differential equations. Details about the discretization and implementation are explained. The algorithms have been intensively tested on synthetic and real images. The advantages of the proposed methods are demonstrated by these experiments.

1 Introduction

For a digital image, inpainting refers to the process of filling-in missing data. It ranges from removing objects from an image to repairing damaged images and photographs.

The term of "digital inpainting" seems to have been introduced into image processing by Bertalmio, Sapiro, Caselles and Ballester [2]. In the past few years, several different approaches have been proposed to tackle this complicated image processing task. The basic idea for most of the inpainting techniques is to do a smooth propagation of the information in the region surrounding the inpainting area and interpolating level curves in a proper way [2, 21, 6]. However, there are different strategies to achieve these goals. In [2], the authors proposed to minimize an energy to compute the restored image and this results in the solving of coupled nonlinear differential equations. In a related work [4], this idea was further extended to guarantee that the level curves are propagated into the inpainting domain. In [3], a connection between the isophote direction of the image and the Navier-Stokes equation was observed and they proposed to solve transport equations to fill in the inpainting domain. This is related to our

*We acknowledge support from the Institute of Computational Mathematics and Scientific Computing of the Chinese Academy of Sciences, the NIH through grant U54RR021813 and the NSF through grants DMS-0312222, ACI-0321917 and DMI-0327077.

†Department of Mathematics, University of Bergen, Johannes Brunsgate 12, N-5007 Bergen, Norway. Email: Tai@mi.uib.no. URL: http://www.mi.uib.no/~tai.

‡Department of Mathematics UCLA, California, USA (e-mail: sjo@math.ucla.edu)

§Department of Mathematics, University of Bergen, Johannes Brunsgate 12, N-5007 Bergen, Norway.

method. Another related work is [11] where a minimization of the divergence is done to construct optical flow functions.

The work of [9, 7] minimizes the TV-norm of the reconstructed image to fill in the missing data. In later work [8, 10], energy involving the curvature of the level curves is used and this is in some sense trying to guarantee that the level curves are connected in a smooth fashion. The equations obtained from such models are highly nonlinear and of higher (fourth) order.

Recently, texture inpainting has attracted attention. In [5], the image in the surrounding area is first decomposed into texture and structure and then propagated into the inpainting domain in different ways. This idea to decompose texture and structure is also used in [12]. Some statistical approaches are used in [1] to do texture synthesis and structure propagation.

We may also mention some recent works which related the phase-field model and Ginzburg-Landau equation to image processing, [15, 16, 13, 12]. These ideas were used in [15, 16, 13] for image segmentation. In [12] they were used for image inpainting.

The idea used in this work was motivated by [19, 20, 2, 3]. We still follow the basic ideas of image inpainting, i.e. we are trying to propagate the information into the inpainting domain along the isophote directions. However, we choose a two-step method to carry out this task as in [20]. The first step involves trying to reconstruct the isophote directions for the missing data. The second step tries to construct an image fitting the restored directions. This is the same idea used in [20] to remove noise from digital images. One new idea which is essential to the present method is that we impose the zero divergence condition on the constructed directions. This guarantees that there exists an image such that its isophote directions are the restored vectors. This is important when the inpainting region is relatively large. In contrast to [3], we obtain our TV-Stokes equation from this consideration which implies that the obtained vectors have the smallest TV-norm. The solution of the Stokes equation will generally not have such a property. We also propose some novel ideas to modify the boundary condition for the inpainting domain to select the information that is propagated into the region. We have only tested our algorithms on propagated structure information. It is possible to combine it with texture inpainting as in [5].

This work is organized as follows. In section 2, we explain the detailed mathematical principles for our methods. First, some geometrical motivation is presented. These geometrical observations are then combined with energy minimization models to get the nonlinear equations which give our inpainting methods. Discretization and implementation details are then supplied. When solving the equations, it is rather easy to change the boundary conditions. Due to this flexibility, we show that it is rather easy to block some information from propagating into the inpainting region. Numerical experiments on real and synthetic images are supplied in Section 3 and comparisons with other methods are discussed.

2 The Mathematical principles

Suppose that an image $u_0 : R \mapsto [a, b]$ is defined on a rectangle domain R. We shall assume that $\Omega \subset R$ is the domain where the data is missing. We want to fill in the information on Ω based in the geometrical and photometric

information surrounding the region Ω. As in [2], we shall use information in a band B around the domain Ω. We shall use $\tilde{\Omega} = \Omega \cup B$ in the following.

2.1 Connection between digital images and flow fields

In [3], the connection between image inpainting and fluid dynamics is done by observing that the isophote directions of an image correspond to an incompressible velocity field. This same observation will be used here in our work. However, the equation we shall use for the inpainting is different and is related to the work of [20]. We give a brief outline of the idea of [20] in the following.

Given scalar functions u and v, denote:

$$\nabla u = (u_x, u_y),\ \nabla^{\perp} u = (-u_y, u_x),\ \nabla \times (u, v) = u_y - v_x,\ \nabla \cdot (u, v) = u_x + v_y.$$

Given an image d_0, the level curves:

$$\Gamma(c) = \{x :\ d_0(x) = c, \quad \forall c \in (-\infty, \infty)\}.$$

have normal vectors $\vec{n}(x)$ and tangential vectors $\vec{\tau}(x)$ given by

$$\vec{n}(x) = \nabla d_0(x) \quad \vec{\tau}(x) = \nabla^{\perp} d_0(x).$$

The vector fields $\vec{n}$ and $\vec{\tau}$ satisfy

$$\nabla \times \vec{n}(x) = 0, \quad \nabla \cdot \vec{\tau}(x) = 0. \tag{1}$$

Suppose that the surface $d_0(x)$ is exposed to rain, then the rain will flow down the surface along the directions $-\vec{n}(x)$. One observation is that the surface d_0 can be constructed from the vector fields $\vec{n}(x)$ or $\vec{\tau}(x)$.

For image inpainting, the information of d_0 in the surrounding band B is known. Thus, we also know the normal and tangential vectors of d_0 in B. The main idea to fill in the information in Ω is to propagate the vector field $\vec{n}$ or $\vec{\tau}$ into the interior region Ω. Afterwards, we construct an image in region Ω to fit the computed vectors in Ω.

Define $\vec{\tau}_0 = \nabla^{\perp} d_0$. There are many different ways to propagate the vectors from B into Ω. In [3], incompressible, inviscid Euler equations are used. Here, we shall use an energy minimization model to propagate the vector fields, i.e. we shall solve

$$\min_{\nabla \cdot \vec{\tau} = 0} \int_{\tilde{\Omega}} |\nabla \vec{\tau}| dx + \frac{1}{\epsilon} \int_B |\vec{\tau} - \vec{\tau}_0|^2 dx \tag{2}$$

Above, $\int_{\tilde{\Omega}} |\nabla \vec{\tau}| dx$ is the total variation for vector field $\vec{\tau}$. We require $\nabla \cdot \vec{\tau} = 0$ to guarantee that the reconstructed vector field $\vec{\tau}$ is a tangential vector for the level curves of a scalar function in the region $\tilde{\Omega}$. The penalization parameter ϵ is chosen to be very small to guarantee that $\vec{\tau} \approx \vec{\tau}_0$ in B. For most of the cases we have tested, it is enough to take B to be just one pixel wide around Ω. For such a case, we can take $\epsilon \to 0$ and thus the minimization problem reduces to find a $\vec{\tau}$ such that $\vec{\tau} = \vec{\tau}_0$ on $\partial\Omega$ which solves:

$$\min_{\nabla \cdot \vec{\tau} = 0} \int_{\Omega} |\nabla \vec{\tau}| dx. \tag{3}$$

We use the total variation norm of $\vec{\tau}$ (as usual in this subject) because the boundary value $\vec{\tau}_0$ may have discontinuities. In order to propagate such a discontinuity into the region Ω, we need to allow $\vec{\tau}$ to have discontinuities and thus the TV-norm is preferred to e.g., the H^1-norm.

We use χ_B to denote the characteristic function over the domain B, i.e. $\chi_B = 1$ in B and $\chi_B = 0$ elsewhere. If we use a Lagrange multiplier λ to deal with the divergence constraint $\nabla \cdot \vec{\tau} = 0$, the Euler-Lagrange equation of (2) is:

$$\begin{cases} -\nabla \cdot \left(\dfrac{\nabla \vec{\tau}}{|\nabla \vec{\tau}|} \right) + \dfrac{\chi_B}{\epsilon}(\vec{\tau} - \vec{\tau}_0) - \nabla \lambda & = 0 \text{ in } \tilde{\Omega}, \\ \nabla \cdot \vec{\tau} & = 0 \text{ in } \tilde{\Omega}, \\ \nabla \vec{\tau} \cdot \vec{\nu} & = \vec{0} \text{ on } \partial\tilde{\Omega}. \end{cases} \tag{4}$$

Here, $\vec{\nu}$ denotes the outer unit normal vector of $\partial\tilde{\Omega}$. Similarly, the Euler-Lagrange equation of (3) is:

$$\begin{cases} -\nabla \cdot \left(\dfrac{\nabla \vec{\tau}}{|\nabla \vec{\tau}|} \right) - \nabla \lambda & = 0 \text{ in } \Omega, \\ \nabla \cdot \vec{\tau} & = 0 \text{ in } \Omega, \\ \vec{\tau} & = \vec{\tau}_0 \text{ on } \partial\Omega. \end{cases} \tag{5}$$

Once the tangential vector field $\vec{\tau}$ is available in $\tilde{\Omega}$, it is easy to obtain the normal vector field $\vec{n}$. Let u and v be the two components of the vector field $\vec{\tau}$, i.e. $\vec{\tau} = (u, v)$. Then, we have

$$\vec{n}(x) = \vec{\tau}^{\perp}(x) = (-v, u). \tag{6}$$

From the vector field $\vec{n}(x)$, we use the same idea as in [20, 2] to construct an image d whose normal vectors shall fit the computed vectors $\vec{n}(x)$. This is achieved by solving the following minimization problem:

$$\min \int_{\tilde{\Omega}} |\nabla d| - \nabla d \cdot \frac{\vec{n}}{|\vec{n}|} dx + \frac{1}{\epsilon} \int_B |d - d_0|^2 dx. \tag{7}$$

The penalization parameter ϵ can be chosen to be same as in (2). Or it can be chosen to be different. In case that B is only one pixel wide around Ω, the above minimization problem reduces to the following problem if we take $\epsilon \to 0$:

$$\min_d \int_{\Omega} |\nabla d| - \nabla d \cdot \frac{\vec{n}}{|\vec{n}|} dx \quad \text{and } d = d_0 \text{ on } \partial\Omega. \tag{8}$$

The Euler-Lagrange equation of (7) is:

$$\begin{cases} -\nabla \cdot \left(\dfrac{\nabla d}{|\nabla d|} - \dfrac{\vec{n}}{|\vec{n}|} \right) + \dfrac{\chi_B}{\epsilon}(d - d_0) & = 0 \text{ in } \tilde{\Omega}, \\ (\dfrac{\nabla \vec{\tau}}{|\nabla \vec{\tau}|} - \dfrac{\vec{n}}{|\vec{n}|}) \cdot \vec{\nu} & = \vec{0} \text{ on } \partial\tilde{\Omega}. \end{cases} \tag{9}$$

Similarly, the Euler-Lagrange equation of (8) is:

$$\begin{cases} -\nabla \cdot \left(\dfrac{\nabla d}{|\nabla d|} - \dfrac{\vec{n}}{|\vec{n}|} \right) & = 0 \text{ in } \Omega, \\ d & = d_0 \text{ on } \partial\Omega. \end{cases} \tag{10}$$

2.2 Discretization

We now explain some of the details in discretizing the equations derived in the last section for numerical simulations. For clarity, we shall only outline the details for algorithms (5) and (10). The discretization for (4) and (9) can be done in a similar way.

For simplicity, the gradient descent method will be used in our simulations. The gradient flow equation for $\vec{\tau}$ is:

$$\frac{\partial \vec{\tau}}{\partial t} - \nabla \cdot (\frac{\nabla \vec{\tau}}{\|\nabla \vec{\tau}\|}) - \nabla \lambda = 0 \qquad \text{in } \Omega, \tag{11}$$

$$\nabla \cdot \vec{\tau} = 0, \text{ in } \Omega, \quad \vec{\tau} = \vec{\tau}_0 \qquad \text{on } \partial\Omega. \tag{12}$$

where $\|\nabla \vec{\tau}\| = \sqrt{|u_x|^2 + |u_y|^2 + |v_x|^2 + |v_y|^2}$. We have tried two algorithms to solve (11)-(12). The first algorithm uses the following iterative procedure to update $\vec{\tau}$ and λ with the time step Δt_1 and initial values properly chosen:

$$\vec{\tau}^{n+1} = \vec{\tau}^n + \Delta t_1 \left[\nabla \cdot \left(\frac{\nabla \vec{\tau}^n}{\|\nabla \vec{\tau}^n\|} \right) + \nabla \lambda^n \right], \tag{13}$$

$$\lambda^{n+1} = \lambda^n + \Delta t_1 \nabla \cdot \vec{\tau}^n \tag{14}$$

The second algorithm updates $\vec{\tau}$ and λ by:

$$\vec{\tau}^{n+1} = \vec{\tau}^n + \Delta t_1 \left[\nabla \cdot \left(\frac{\nabla \vec{\tau}^n}{\|\nabla \vec{\tau}^n\|} \right) + \nabla \lambda^n \right], \tag{15}$$

$$-\Delta \lambda^{n+1} = \nabla \cdot \left[\nabla \cdot \left(\frac{\nabla \vec{\tau}^n}{\|\nabla \vec{\tau}^n\|} \right) \right]. \tag{16}$$

In (16), Δ denotes the Laplace operator and we impose a zero Neumann boundary condition for λ^{n+1}. If $\nabla \cdot \tau^0 = 0$ and (16) is satisfied by all λ^n, then we see from (15) that

$$\nabla \cdot \tau^{n+1} = 0, \quad \forall n.$$

We use a staggered grid to approximate u, v and λ. Note that $\vec{\tau} = (u, v)$ is used to construct d. When we try to compute d from (9) or (10), we are trying to enforce the following relation approximately: $u = -d_y, \quad v = d_x$. Due to this relation, the grid points used in the approximation for u are chosen to be the points marked with $*$, see Figure 1. The approximation points for v are marked with $\circ$. The centers of the rectangle elements marked with $\star$ are used as the approximation points for λ. The vertices of the rectangular mesh are used as the approximation points for d. The horizontal axis represents the x-variable and the vertical axis represents the y-variable, c.f Figure 1.

For a given domain Ω, we use $U_h(\Omega)$ to denote all the approximation points $*$ for u inside Ω, $V_h(\Omega)$ to denote all the approximation points $\circ$ for v inside Ω, $\Lambda_h(\Omega)$ to denote all the approximation points $\star$ for λ inside Ω and $D_h(\Omega)$ to denote all the approximation points for d inside Ω. The updating formulae for

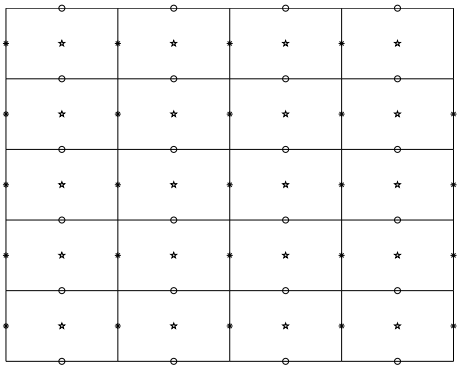


Figure 1: The pixels and the approximation points for u, v, λ and d. The approximation points are: $*$ for u, $\circ$ for v, $\star$ for λ.

(u, v) and λ for (13)-(14) are:

$$u^{n+1} = u^n + \Delta t_1 \left(D_x^- \left(\frac{D_x^+ u^n}{T_1^n} \right) + D_y^- \left(\frac{D_y^+ u^n}{T_2^n} \right) + C_x^{h/2} \lambda^n \right) \quad \text{on } U_h(\Omega), \tag{17}$$

$$v^{n+1} = v^n + \Delta t_1 \left(D_x^- \left(\frac{D_x^+ v^n}{T_2^n} \right) + D_y^- \left(\frac{D_y^+ v^n}{T_1^n} \right) + C_y^{h/2} \lambda^n \right) \quad \text{on } V_h(\Omega), \tag{18}$$

$$\lambda^{n+1} = \lambda^n + \Delta t_1 (C_x^{h/2} u^{n+1} + C_y^{h/2} v^{n+1}) \quad \text{on } \Lambda_h(\Omega). \tag{19}$$

Above, $D_x^{\pm}, D_y^{\pm}$ are the standard forward/backward finite difference operators and $C_x^{h/2}, C_x^{h/2}$ are the central finite difference operators with mesh size $h/2$. h denotes the mesh size for the approximations and is taken to be one. The terms T_1^n and T_2^n are evaluated as in the following:

$$T_1^n = \sqrt{|D_x^+ u|^2 + |C_y^h u|^2 + |C_y^h v|^2 + |D_y^+ v|^2 + \epsilon} \text{ on } \Lambda_h(\Omega), \tag{20}$$

$$T_2^n = \sqrt{|C_x^h u|^2 + |D_y^+ u|^2 + |D_y^+ v|^2 + |C_y^h v|^2 + \epsilon} \text{ on } D_h(\Omega). \tag{21}$$

If we use the second algorithm to compute (u, v) and λ from (15)-(16), the solution of (16) is not unique due to the use of the Neumann boundary condition. We fix the value of λ to be zero at one point on the boundary to overcome this problem, which is standard for this kind of problem. Fast methods, like the FFT (Fast Fourier Transformation), can be used to solve (16).

Once the iterations for u and v have converged to a steady state, we use them to obtain d. Note that the relation between (u, v) and $\vec{n}$ is as in (6). Similar as in [20], the following gradient flow scheme is used to update d of (10):

$$d^{n+1} = d^n + \Delta t_2 \Bigg(\left(D_x^- \left(\frac{D_x^+ d^n}{D_1^n} + \frac{v}{\sqrt{\hat{u}^2 + v^2 + \epsilon}} \right) + \left(D_y^- \left(\frac{D_y^+ d^n}{D_2^n} - \frac{u}{\sqrt{u^2 + \hat{v}^2 + \epsilon}} \right) \right) \text{ on } D_h(\Omega). \tag{22}$$

In the above, $\hat{u}, \hat{v}$ are the average values of the four nearest approximation points and

$$D_1^n = \sqrt{|D_x^+ d^n|^2 + |C_y^h d^n|^2 + \epsilon} \text{ on } D_h(\Omega), \tag{23}$$

$$D_2^n = \sqrt{|C_x^h d^n|^2 + |D_y^+ d^n|^2 + \epsilon} \text{ on } D_h(\Omega). \tag{24}$$

This iteration is the standard gradient updating for d. We could use the AOS scheme of [17, 18] to accelerate the convergence. The AOS scheme was first proposed in [17, 18]. It was later rediscovered in [22, 14] and used for image processing problems.

Up to now we have only explained the approximation details for (5) and (10). It is easy to see that the discretization for (4) and (9) can be done in a similar way. The Dirichlet or Neumann boundary conditions for the different equations are implemented in the standard way and we will omit the details.

2.3 Other kind of boundary conditions

We have proposed two alternatives to deal with the information which is in the surrounding area of Ω, i.e.

- Using information in a narrow band around the inpainting region Ω and trying to propagate this information into the region Ω using equations (4) and (9).
- Using information of the two nearest pixels around the inpainting region Ω and using equations (5) and (10) to propagate the information into the region Ω.

There is no strong evidence about which of these two alternatives is better. In fact, numerical experiments show that this is image dependent. In most of the tests given in this work, we have used the boundary conditions (5) and (10).

In the following, we shall even propose another boundary condition to treat some special situations. For some images, we may want some of the information from the surrounding area to be propagated into Ω, while some other information from the surrounding area is not welcome to be so propagated, see Figures 9, 11, 12. In order to deal with this kind of situation, we propose the following alternative:

- Decompose the boundary $\partial\Omega$ into two parts, i.e. $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$. For equation (5), replace the boundary condition by

$$a)\ \vec{\tau} = \vec{\tau}_0 \text{ on } \partial\Omega_D, \quad b)\ \vec{\tau} = \vec{0} \text{ on } \partial\Omega_N, \tag{25}$$

  and replace the boundary condition of (10) by

$$a)\ d = d_0 \text{ on } \partial\Omega_D \quad b)\ \frac{\partial d}{\partial \vec{\nu}} = 0 \text{ on } \partial\Omega_N. \tag{26}$$

Condition (26.b) means that we do not want to propagate any information through $\partial\Omega_N$. Due to the fact that $\nabla d^\perp \approx \vec{\tau}$, condition (26.b) implies that we must have condition (25.b) for $\vec{\tau}$ on $\partial\Omega_N$. A similar procedure can be performed for equations (4) and (9).

3 Numerical Experiments

First, we explain how to choose ε, Δt_1 and Δt_2 in numerical implementations. We add ε to the denominator to avoid dividing by zero in (20)-(21) and (23)-(24). If ϵ is chosen to be large, the computed image will be smoothed a bit. If ϵ is chosen to be too small, it may slow down the convergence. We have chosen ϵ to be the same in (20)-(21) and (23)-(24), but it will differ from example to example.

With large Δt_1 and Δt_2, the iterations will converge faster, but if they are too large, the scheme is unstable. For most experiments $\Delta t_1 \approx 0.03$ will lead to convergence of the normal vectors. A smaller Δt_1 will also work, but more iterations might be necessary. If the normal vectors are smooth, Δt_2 is less sensitive and can be chosen to be large. If the vector field is less smooth, Δt_2 must be smaller.

Example 1

In this example we test out our method on an image from a Norwegian newspaper. The image shows a man jumping from Jin Mao Tower, a 421 meter tall building in the city of Shanghai. We want to remove the man and restore the city in the background.

The first part of the code computes the normal vectors in the missing region. From Figure 3 we see that the vectors are propagating into the inpainting region in a smooth fashion. When $\Delta t_1 = 0.03$ and $\epsilon = 10$ are used, a steady state is reached after 3000 iterations using (13)-(14). If we use (15)-(16), less than 1000 iterations are needed to reach a steady state, see Figure 3 e) and Figure 3 f).

The second part reconstructs the image using the computed normal vectors. Figure 4 shows how the man is gradually disappearing during the iterations. With $\Delta t_2 = 0.15$ it takes 30000 iterations before a steady state is reached. In the resulting image the man has disappeared completely and the background is restored in a natural way. There are no discontinuities in the sky, and the skyline is almost a straight line. It is nearly impossible to detect that the sky and the skyline contains the missing region.

Figure 2: The original image.

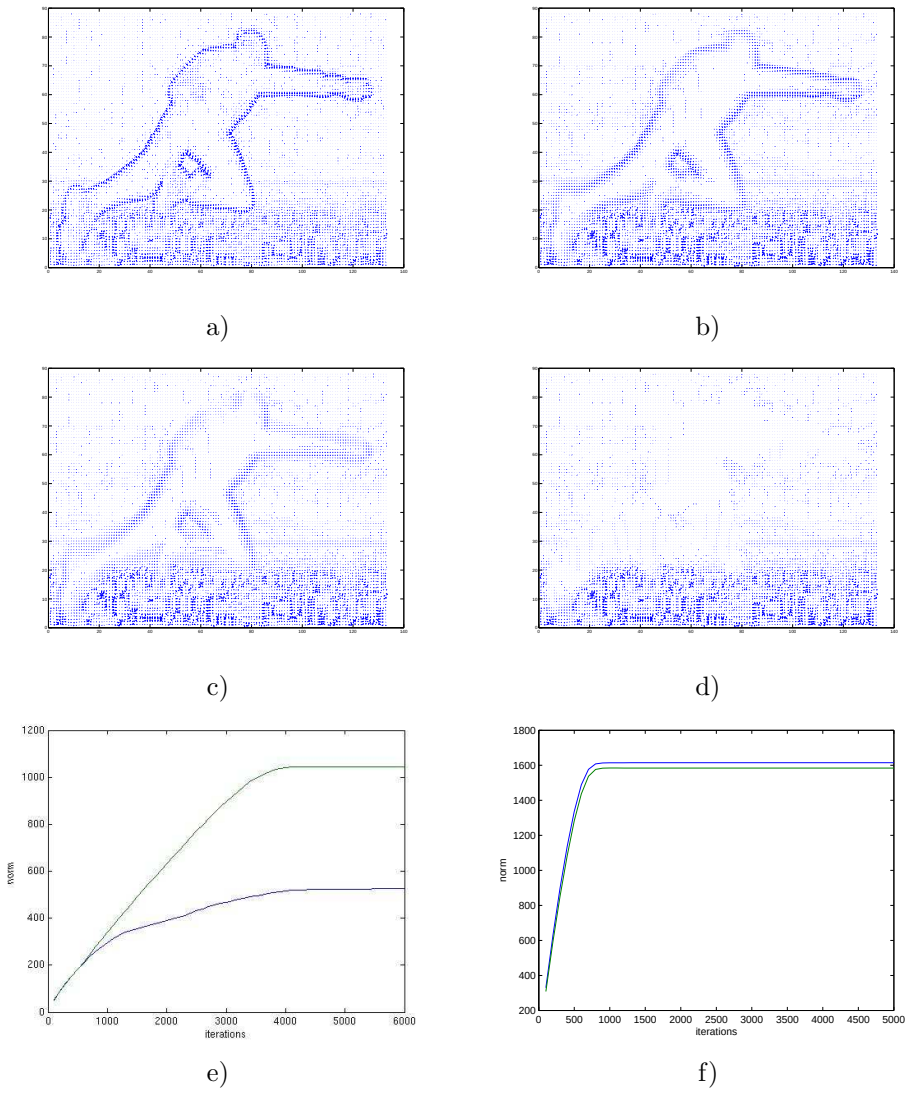


Figure 3: The restored flow vector $\vec{\tau}$ using (13)-(14) at different iterations. a) at iteration 0; b) at iteration 1000; c) at iteration 2000; d) at iteration 3000; e) The plot for $\|u\|$ and $\|v\|$ which shows that the equations (13)-(14) reach a steady state, i.e. at iteration 3000. f) In this plot, we show the convergence for $\|u\|$ and $\|v\|$ using equations (15)-(16). They reach steady states quicker than (13)-(14), i.e. at iteration 1000.

Example 2

We test our method on some well-know examples which have been tested by others using different methods [2]. We use these results to show the quality of the restored images compared with other methods.

In the example shown in Figure 5, red text is written over the picture. The

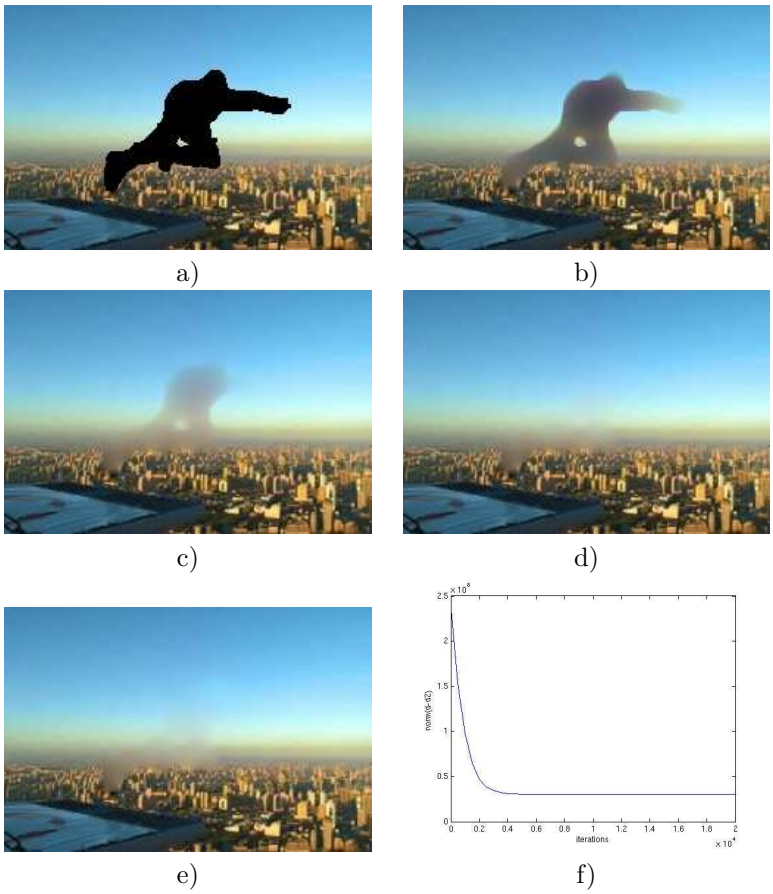


Figure 4: The restored image d using equation (10) at different iterations. a) at iteration 0; b) at iteration 10000; c) at iteration 20000; d) at iteration 30000; e) The restored image using the new method (15)-(16) to find $\vec{\tau}$. f) The plot for $\|d - d_0\|$ which shows that the equation (5) reaches a steady state, i.e. at iteration 30000.

text is the inpainting area, and we want to fill it with information from the image. With $\epsilon = 1$ and $\Delta t_1 = 0.03$ the normal vectors converge after 7000 iterations for (13)-(14). The second part of the code converged after only 3000 iterations with $\Delta t_2 = 0.5$.

In Figure 6, another image which has been tested in the literature, is used here to compare our method with the others, [2, 1]. The image has the white text 'Japanese animation', and we want to remove this. An area around the text is lighter than the background and has to be restored as well. Figure 6 b) shows the manually obtained inpainting region. Figure 6 c) shows restored image. The values for Δt_1 and Δt_2 are chosen to be the same as in the previous example, and the convergence is nearly the same.

Figure 7 a) shows an old photo which has been damaged. We mark the inpainting region in white colour, as shown in Figure 7 b) and try to restore it. The result is shown in Figure 7 c).

a)

b)

c)

Figure 5: a) The original image. b) The restored image using equations (5) and (10). c) The difference image.

The image in Figure 8 a) shows another situation where our algorithm can be applied. The image has a piece of musical notes written on it. A large amount of information is lost, but it is scattered on the image in narrow areas. The first part converges after 2500 iterations and the second part converges after 1000 iterations when using our algorithm for this image. The restored image in Figure 8 b) looks rather good.

Example 3

To test the code for the new boundary condition (25)-(26), we created a simple image, see Figure 9. Information is missing in a rectangle in the middle of the

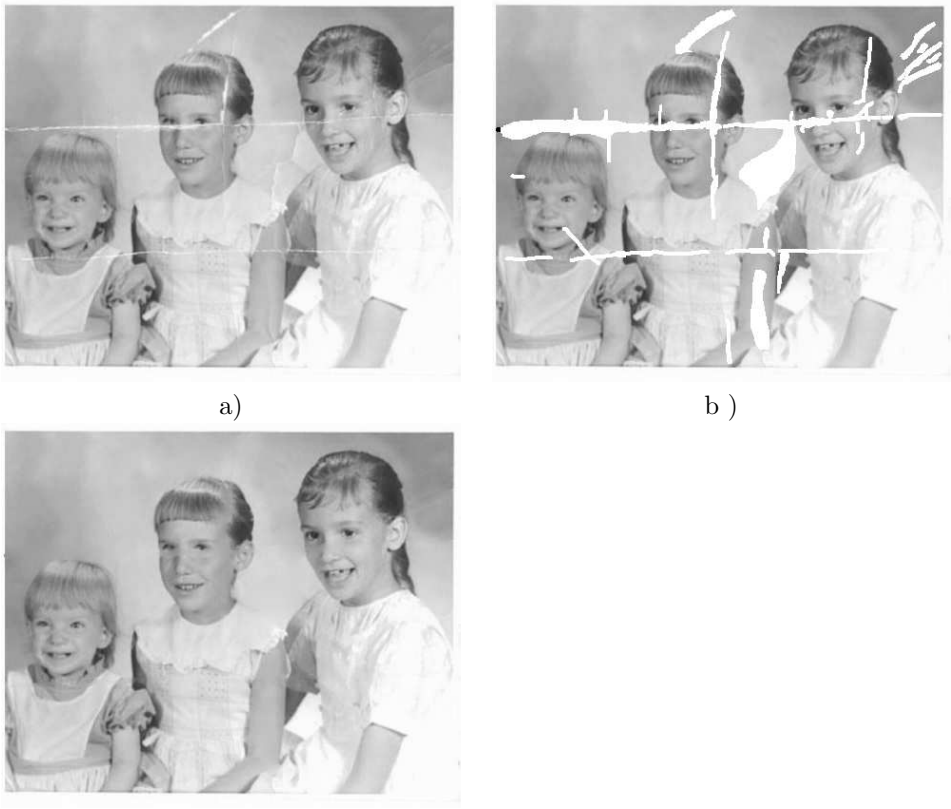
a) b)

c)

Figure 6: a) The original image. b) The image with the inpainting region obtained manually. c) The restored image using equations (5) and (10).

image which only has two intensity values. If we use Dirichlet boundary conditions (5)-(10), all information from the surrounding area will be transported into the inpainting region. If the Neumann boundary is used (25)-(26), it is possible to choose which intensity value to be selected to propagate into the inpainting region. The result is shown in Figure 9. The result using Dirichlet boundary conditions is displayed in Figure 9 b). With ϵ=0.0001, $\Delta t_1 = 0.01$, the normal vectors converged after 12000 iterations and with $\Delta t_2 = 0.2$ the second part converged after 25000 iterations. With a larger ϵ, the corners and the boundary close to the corners may be smeared.

Figure 9 c) shows a similar test with Dirichlet conditions on the upper half and with Neumann boundary conditions on the lower half of the boundary of the inpainting region. From Figure 9 c) we see that only one of the colours was



a) b)

c)

Figure 7: a) The original image d_0. b) The image with the inpainting region white. c) The restored image d.

selected and propagated to the interior.

Example 4

In this example, we process an image from the match between Norway and Croatia in the XIX Men's World Championship. We want to remove the Croatian player in Figure 10. When a Dirichlet condition is used around the whole boundary, Figure 11 a), colours from the Norwegian players propagate into the background. To make it look natural, it is necessary to use Neumann boundary conditions around the two Norwegian players. The inpainting region and the Neumann boundary are marked in Figure 11 b). Figure 11 c) shows the restored image using this new boundary condition. When Neumann boundary condition is used, the colour on the Neumann boundary does not influence the interior.

Example 5

This example has more texture in the background. We want to remove the snowboarder and fill in the missing region. It is not desirable that the yellow object in the front propagates into the inpainting region. Figure 12 d) shows that the best result is obtained with Neumann conditions on part of the boundary.

4 Conclusion

In this work, we have proposed a method which uses two second order equations to do image inpainting. The equations used here are similar to the equations used in [2] and [3]. By imposing the zero divergence condition which was not imposed in [2], it seems that our methods are able to produce better results when the inpainting region is rather large in diameter.

It is an interesting problem to study the existence and uniqueness for the solution for the equations we used. We have observed numerically that the gradient flow equations for (5) and (10) seem to have stable and unique solutions under the condition that the initial values are fixed.

References

[1] Criminisi A., Perez P., and Toyama K. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Image Process.*, 13(9), 2004.

[2] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE Trans. Image Processing*, (10):1200–1211, 2000.

[3] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. Navier-Stokes, fluid dynamics and image and video inpainting. In *Proc. Conf. Comp. Vision Pattern Rec.*, pages 355–362, 2001.

[4] M. Bertalmio, G. Sapiro, C. Ballester, and V. Caselles. Image inpainting. *Computer Graphics, SIGGRAPH*, 2000.

[5] M. Bertalmio, L. Vese, G. Sapiro, and O. Osher. Simultaneous texture and structure image inpainting. *IEEE Trans. Image Process.*, 10(8), 2003.

[6] Vicent Caselles, Simon Masnou, Jean-Michel Morel, and Catalina Sbert. Image interpolation. In *Séminaire sur les Équations aux Dérivées Partielles, 1997–1998*, pages Exp. No. XII, 15. École Polytech., Palaiseau, 1998.

[7] Tony Chan and Jianhong Shen. Variational restoration of nonflat image features: Models and algorithms. *SIAM J. Appl. Math.*, 61(4):1338–1361, 2000.

[8] Tony F. Chan, Sung Ha Kang, and Jianhong Shen. Euler's elastica and curvature-based inpainting. *SIAM J. Appl. Math.*, 63(2):564–592, 2002.

[9] Tony F. Chan and Jianhong Shen. Mathematical models for local nontexture inpaintings. *SIAM J. Appl. Math.*, 62(3):1019–1043, 2002.

[10] Tony F. Chan, Jianhong Shen, and Luminita Vese. Variational PDE models in image processing. *Notices Am. Math. Soc.*, 50(1):14–26, 2003.

[11] F. Guichard and L. Rudin. Accurate estimation of discontinuous optical flow by minimizing divergence related functionals. *Proceedings of International Conference on Image Processing, Lausanne, September, 1996*, pages 497–500, 1996.

[12] Grossauer H. and Scherzer O. Using the complex Ginzburg-Landau equation for digital inpainting in 2D and 3D. In *Sacle space method in computer vision*, Lectures notes in Computer Sciences 2695. Springer, 2003.

[13] Shen J. Gamma-convergence approximation to piecewise constant mumford-shah segmentation. Tech. rep. (05-16), UCLA, Applied mathematics, 2005.

[14] Weickert J. *Anisotropic Diffusion in Image Processing.* Stuttgart, B. G. Teubner, 1998.

[15] Johan Lie, Marius Lysaker, and Xue-Cheng Tai. A binary level set model and some applications to image processing. *IEEE Trans. Image Processing*, to appear.

[16] Johan Lie, Marius Lysaker, and Xue-Cheng Tai. A variant of the levelset method and applications to image segmentation. *Math. Comp*, to appear.

[17] T. Lu, P. Neittaanmaki, and X.-C. Tai. A parallel splitting up method and its application to Navier-Stoke equations. *Applied Mathematics Letters*, 4:25–29, 1991.

[18] T. Lu, P. Neittaanmaki, and X.-C. Tai. A parallel splitting up method for partial differential equations and its application to Navier-Stokes equations. *RAIRO Math. Model. and Numer. Anal.*, 26:673–708, 1992.

[19] M. Lysaker, A. Lundervold, and X.-C. Tai. Noise Removal Using Fourth-Order Partial Differential Equation with Applications to Medical Magnetic Resonance Images in Space and Time. *IEEE Transactions on Image Processing*, 12(12):1579–1590, 2003.

[20] M. Lysaker, Stanley Osher, and Xue-Cheng Tai. Noise removal using smoothed normals and surface fitting. *IEEE Trans. Image Processing*, 13(10):1345–1457, 2004.

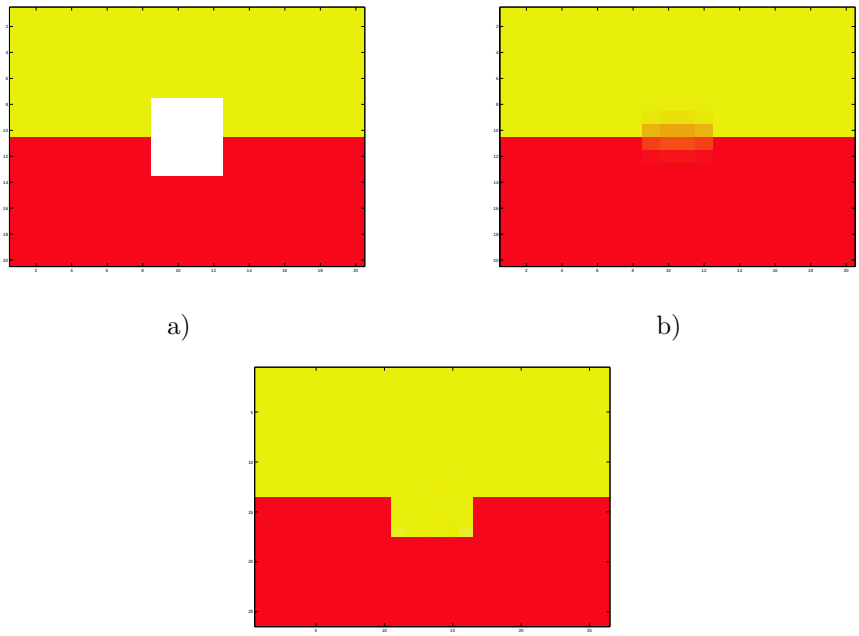
[21] Simon Masnou. Disocclusion: a variational approach using level lines. *IEEE Trans. Image Process.*, 11(2):68–76, 2002.

[22] J. Weickert, B. H. Romeny, and M. A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Trans. Image Process.*, 7:398–409, 1998.

a)

b)

Figure 8: a) The image with the inpainting region white. b) The restored image using equations (5) and (10).



a) b)

c)

Figure 9: a) The image with the inpainting region marked. b) The image obtained with Dirichlet boundary. c) The image obtained using Dirichlet and Neumann boundary conditions.

Figure 10: An image from the match between Norway and Croatia in the XIX Men's World Championship.

Figure 11: a) The restored image using Dirichlet boundary conditions. b) The image with the inpainting region violet. c) The restored image using Dirichlet and Neumann boundary conditions.

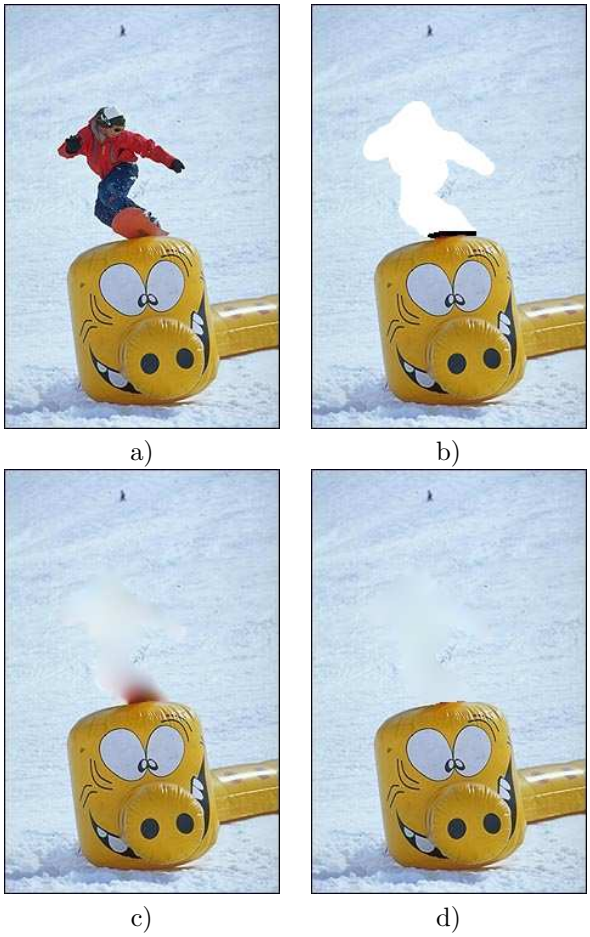


Figure 12: a) A photo taken by Espen Lystad, a well-known snowboard photographer in Norway. b) The image with the inpainting region marked. The Neumann boundary is black. c) The restored image only using Dirichlet boundary condition. d) The restored image using Dirichlet and Neumann boundary conditions.