Image Inpainting with the Navier-Stokes Equations

Wilson Au, wilsona@sfu.ca **Ryo Takei,** rrtakei@sfu.ca

Final Report, APMA 930

Abstract

Image inpaining "involves filling in part of an image (or video) using information for the surrounding area"([1]). This report summarizes an application of numerical solutions to the Navier-Stokes equations for this well-studied image processing problem. It exploits the remarkable relationship between the steady state solution of the streamfunction in fluid flow and the (grayscale) image intensity level in image processing.
This report is a follow-up of an approach first discovered in 2001 by A. Bertozzi, et. al in [1].

1 Preliminaries and Motivation

Image inpainting (from hereon, simply *inpainting*) is the technique of filling in a region of an image based on the information outside the region. The distinction between inpainting and *denoising* should be made clear: deblurring generally attemps to modify regions that are individual pixels, while inpainting involves modifying a larger area. Applications for inpainting are generally to remove unwanted patterns in photos, from scratches and vandalization to superimposed letters.
Unless otherwise stated, an image will refer to a grayscale image.

1.1 Grayscale Image Basics

A digital grayscale image, I, is an $m \times n$ matrix, where at each index, $I_{i,j}$ consists of an integer value from 0 to 255 (we will only consider rectangular images). The (i, j)th index in I is equivalent to the *pixel* at the corresponding location. This value is referred to as the *graylevel* at location (i, j), where 0 corresponds to pure black, 255 to pure white, and all intermediate values to different shades of gray. We let D be the set of indicies $(i, j) \in \{1, 2, \ldots, n\} \times \{1, 2, \ldots, m\}$ where I is defined.
Throughout this report, however, we equivalently treat I as a function from a discrete domain of indicies $\{1, 2, \ldots, n\} \times \{1, 2, \ldots, m\}$ to the integers mod 256.

1.2 The Inpainting Problem

Suppose we have a subset $\Omega \subset D$ where we would like to modify the graylevel of I based on the information of I from the surrounding region $D\backslash\Omega$. Ω will be referred to as the *inpainting region*. The modified image, which we call I^* or the *solution*, will, by definition, have equal grayvalues as I in $D\backslash\Omega$. The process of finding the appropriate I^*, we call, the *inpainting problem* (See Figure 1).

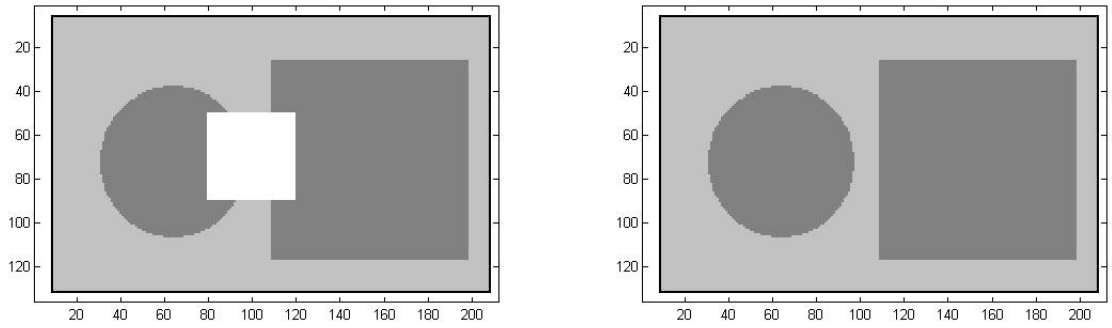

Figure 1: *An example of Inpainting.* Left: The rectangular region in the center in white is Ω, the inpainting region. Right: A 'good' solution to the inpainting problem; the grayvalues in Ω are reasonable with respect to the grayvalues outside of Ω.

Generally in applications, the ratio $|\Omega|/|D|$ is approximately $10 \sim 20\%$ ([3]). This means that we have ample information in $D\backslash\Omega$ to base our guess for I^* in Ω. Also, Ω need not be rectangular nor connected, as we will see later.

There are already several algorithms available as commercial software that perform inpainting. Some prompt the user for parameters and/or manual adjustment in completing the inpainting process. The algorithm we have applied is robust in the sense that no user input is required other than the image I and the inpainting region Ω. Note that the algorithm does not find the appropriate Ω; this would involve techniques for other areas in image processing, such as *image segmentation*.

In some sense, inpainting is an art form; There is never *the* correct solution, but certainly some solutions are more appealing than others. The goal in inpainting is to derive the most appealing solution in the region to be filled in. The mathematical equivalent to an appealing solution will be motivated later.

1.3 The Solution Criterion

The most natural approach to solve the inpainting problem is to mimic how professional image restorators inpaint manually. As discussed in [2], restorators extend edges from the boundary of Ω, connect these extended edges, and then fill in intra-region accordingly. This idea has been proven to produce satisfactory, if not remarkable, solutions ([1],[2],[3]). To

motivate this mathematically, we must introduce several more key notions.
First, *isophotes* are level lines of equal graylevels. Mathematically, the direction of the isophotes can be interpreted as

$$\nabla^{\perp} I,$$

where $\nabla^{\perp} = (-\partial_y, \partial_x)$, the direction of the smallest change. Next, the *smoothness* can be interpreted as,

$$\Delta I,$$

where Δ is the usual Laplacian operator. Notice that, misleading as it may seem, $|\Delta I|$ is large for non-smooth regions, while ≈ 0 in smooth regions. Nevertheless, this is a good way of quantifying the smoothness of an image at a particular location. Generally, ΔI will extract edges (and noise) in an image.
In light of $\nabla^{\perp} I$ and ΔI, in order to mimic the idea of image restorators, we conclude:

> *Propagate ΔI in the direction of $\nabla^{\perp} I$ from the boundary of Ω. When all information is propagated, ie. the level lines of ΔI is parallel to $\nabla^{\perp} I$, we have a solution.*

See Figure 2 below.

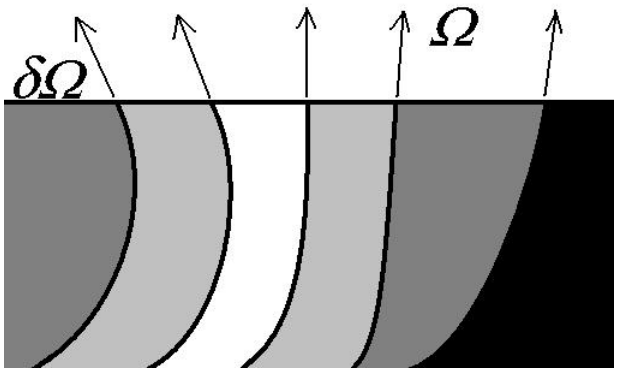

Figure 2: *The inpainting problem as approached by Bertalmio, et. al..* The smoothness information ΔI is propagated in the direction of the isophotes. (The vertical curves represents isophote lines.)

Notice that, unlike the restorators, the graylevels in the intra-regions are automatically determined by the smoothness in Ω and the graylevels on the boundary of Ω. Mathematically, therefore, we have the solution criterion for the inpainting problem:

> *The solution I^* satisfies*
>
> $$\nabla^{\perp} I^* \cdot \nabla \Delta I^* = 0 \qquad (1)$$
>
> *and is equal to I on $\partial\Omega$, the boundary of Ω.*

Bertalmio, et. al. ([2]) iteratively propagates ΔI in the direction of $\nabla^{\perp} I$ until a steady state (1) is met. A simple verification of this steady state condition is shown in Figure 3. How the algorithm introduced in this paper propagates ΔI is described in the following sections.

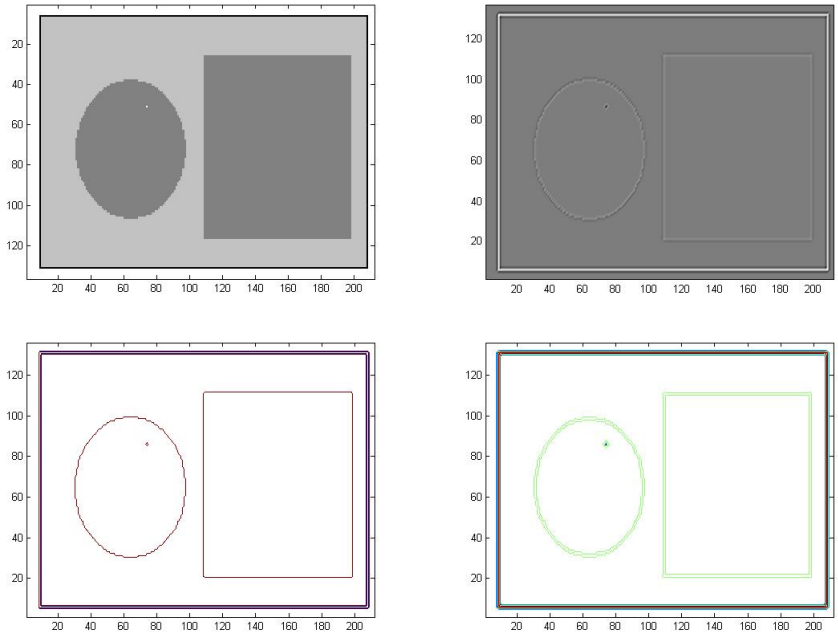

Figure 3: *Verification of the solution criterion on a sample image.* Top Left: The Original Image I. Top right: The smoothness ΔI. Bottom Left: the Isophote lines $\nabla^{\perp} I$. Bottom Right: The level curve of the smoothness ΔI. Note that the isophotes directions are parallel to the level curves of the smoothness, ie. $\nabla^{\perp} I \cdot \nabla \Delta I = 0$.

1.4 Navier-Stokes and the PDE for Inpainting

Incompressible, Newtonian flow obey the Navier-Stokes equations,

$$v_t + v \cdot v = -\nabla p + \nu \Delta v, \quad \nabla \cdot v = 0, \tag{2}$$

where v is the velocity vector, p is the pressure and ν is the viscosity. For 2 dimensional flows, introduce a stream function Ψ, where,

$$\nabla^{\perp} \Psi = v,$$

eliminates p in and identically satisfies the divergence free condition in (2). Letting $\omega = \nabla \times v$, the vorticity, we obtain the vorticity-stream function formulation for the Navier-Stokes equations:

$$\omega_t + v \cdot \nabla \omega = \nu \Delta \omega. \tag{3}$$

In the case of near absence of viscosity, ie. $\nu \approx 0$, we have the steady state solution of (3) approaching,

$$v \cdot \nabla \omega = \nabla^{\perp} \Psi \cdot \nabla \Delta \Psi \approx 0 \tag{4}$$

Notice the remarkable similarity between (4) and the solution criterion (1) for the inpainting problem! Exploiting this fact and replacing Ψ with an image matrix I [1], we summarize the

[1] Ψ is a continuous function defined on a continuous domain, while I is like an integer function defined on a discrete domain. Since we will discretize Ψ using finite difference techniques, the latter discrepancy is resolved. For the former case, values are rounded to the nearest integer.

counterparts between 2D incompressible fluid flow and image inpainting in the table below ([1]).

Fluid dynamics	Image processing
stream function Ψ	Image intensity I
fluid velocity $\vec{v} = \nabla^{\perp}\Psi$	isophote direction $\nabla^{\perp} I$
vorticity $\omega = \Delta\Psi$	smoothness $w = \Delta I$
viscosity ν	anisotropic diffusion ν

In image processing terms, we now have the counterpart to the vorticity-stream function formulation (3):

$$w_t + v \cdot \nabla w = \nu \nabla \cdot (g(|\nabla w|)\nabla w), \tag{5}$$

where $\Delta I = w$, the vorticity, and $\nabla^{\perp} I = v$, the direction of the isophotes.
The g in (5) accounts for *anisotropic diffusion*, or edge-preserving diffusion [2]. In general,

$$g(0) = 1, \lim_{s\to\infty} g(s) = 0, \text{ and monotonically decreasing}$$

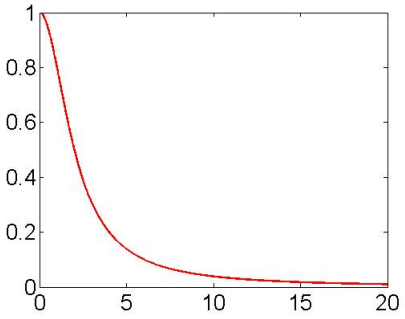

Figure 4: The Perona-Malik anisotropic diffusion function g, with $K = 2$.

Notice how the diffusion term is large for smooth areas while close to zero on edges, and therefore tends to sharpen images ([5]). One example for g is the so-called Perona-Malik anisotropic diffusion ([6], Figure 4), given by,

$$g(s) = \frac{1}{1 + (s/K)^2}, \tag{6}$$

where K is a predefined diffusion parameter.
The goal is to evolve (5) to a steady state.

[2]For simplicity in implementation, engineers sometimes use the somewhat crude approximation $\nabla \cdot (g(|\nabla w|)\nabla w) \approx g(|\nabla w|)\Delta w$. See, for example, [7].

2 Implementation

2.1 The Algorithm

The algorithm exploits the same technique used to solve 2D fluid flow in a rectangular domain.
The two equations, (5) and $\Delta I = w$ must be solved simultaneously. One way to achieve this is to use a standard time-marching technique where we have, at time level $t = n\Delta t$ [3], w^n and I^n for which to be iterated until a steady state is reached.
Assume, for now, that the inpainting region Ω is rectangular. The algorithm solves the coupled equations in Ω, using necessary boundary condition information extracted from $\partial\Omega$.

2.2 The Finite Difference Scheme

In evolving (5), two finite difference approaches were implemented: Forward-time centered-space (FTCS) and forward-time Upwind (FTUp).
Suppose we have an m^* by n^* image I. We treat I to be defined on a continuous domain $[0, n] \times [0, m]$ and discretized into a m^* by n^* grid, with grid spacing $\Delta x = \Delta y = 1$. The FTCS scheme looks as follows:

$$w_{i,j}^{n+1} = w_{i,j}^n + \Delta t \left[-u^n D_x^0 w_{i,j} - v^n D_y^0 w_{i,j} + \nu\{\text{diffusion discretization}\} \right]$$

where D_x^0, D_y^0 denotes the centered approximation of the first derivative; The discretization for the diffusion will be formulated in Section 2.4.
In implementing the FTUp scheme, we note that for a linear advection equation $w_t = aw_x$, the direction of the upwind depends on the wave speed a. If we expand the dot product for the advection term in (5), we have,

$$w_t + uw_x + vw_y = \nu \nabla \cdot (g(|\nabla w|)\nabla w).$$

When discretizing, the direction of upwind for the terms w_x and w_y, therefore, depends on the sign of their coefficients I_y and I_x. Written in concise form, we have the upwind discretization of the advection terms as:

$$\begin{aligned} uw_x &\approx |u^n| \frac{w^n_{i+sgn(u^n),j} - w^n_{i,j}}{\Delta x} \\ vw_y &\approx |v^n| \frac{w^n_{i,j+sgn(v^n)} - w^n_{i,j}}{\Delta y} \end{aligned}$$

The full discretization becomes,

$$\begin{aligned} w_{i,j}^{n+1} = w_{i,j}^n + \Delta t(&-|u^n| \frac{w^n_{i+sgn(u^n),j} - w^n_{i,j}}{\Delta x} - |v^n| \frac{w^n_{i,j+sgn(v^n)} - w^n_{i,j}}{\Delta y} \\ &+\nu\{\text{diffusion discretization}\}) \end{aligned}$$

[3] The reader should note the subtle abuse of notation for Δ as a Laplacian operator and the "difference" operators used for grid spacing, etc.

The reason for choosing an explicit scheme is clear; Since one must evolve in time and solve the poisson equation at each time-step, implementation of an implicit scheme will cause difficulty in implementation.

In solving the poisson equation $\Delta I^n = w^n$, we first note that the standard centered discretization of the Laplacian becomes,

$$\Delta y^2(I_{i+1,j} + I_{i-1,j}) + \Delta x^2(I_{i,j+1} - I_{i,j-1}) - 2I_{i,j}(\Delta y^2 + \Delta x^2) = \Delta x^2 \Delta y^2 w^n_{i,j}$$

Then, reshaping I^n and w^n into vectors of length m^*n^*, $\vec{I}^n$ and $\vec{w}^n$, yields a linear system of the form $A\vec{I}^n = \vec{w}^n$, where A is a sparse, $m^*n^* \times m^*n^*$ matrix (for boundary conditions see Section 2.3). Techniques for solving linear systems are well-studied and hense a wide variety of methods are available. For example, [1] uses the Jacobi method, while our algorithm uses the successive over-relaxation (SOR) method [4].

2.3 Boundary Conditions

For the finite differences schemes derived in Section 2.2, boundary conditions for both I^n and w^n are required. The implementation of these boundary conditions are slightly different from fluid flows ([8]), since, I, the counterpart to the stream function Ψ is the only known information on $\partial\Omega$.

The boundary condition for solving the poisson equation $\Delta I^n = w^n$ is clear as one can simply use the dirichlet conditions with the values of I on $\partial\Omega$.

In computing the dirichlet conditions for $w = \Delta I$ on $\partial\Omega$, one requires to compute several one-sided approximation of derivatives. As mentioned in Section 1.2, we use the fact that I is well known outside Ω, not just on the boundary $\partial\Omega$. Computing w on the boundary requires several steps:

1. *Compute u and v for three-pixel thickness on $\partial\Omega$.* We have $u = -I_y$ and $v = I_x$; using centered difference whenever possible and one-sided three-point formulae otherwise, compute u and v from I outside Ω. In total, we require a three-pixel thickness of u and v outside Ω; the reason for this will be clear.

2. *Compute u_y and v_x for one-pixel thickness on $\partial\Omega$.* Using values of u and v computed in step 1, computed using centered difference whenever possible and one-sided three-point formulae otherwise the values for u_y and v_x for one-pixel thickness.

3. *Compute w (for one-pixel thickness) on $\partial\Omega$.* From values computes in step 2, take the difference $w = -u_y + v_x$. We now have the dirichlet boundary condition for w.

By the one-sided three-point formulae for the first derivative, we mean:

$$\begin{aligned} f'_j &= \frac{-f_{j+2} + 4f_{j+1} - 3f_j}{2\Delta x} + O(\Delta x^2) \\ f'_j &= \frac{3f_j - 4f_{j-1} + f_{j-2}}{2\Delta x} + O(\Delta x^2) \end{aligned}$$

These formulae for the derivative, along with the centered difference formulae maintains a second-order accuracy on the boundary conditions. See Figure 5.

[4] For testing purposes on MATLAB, the infamous '\' was used.

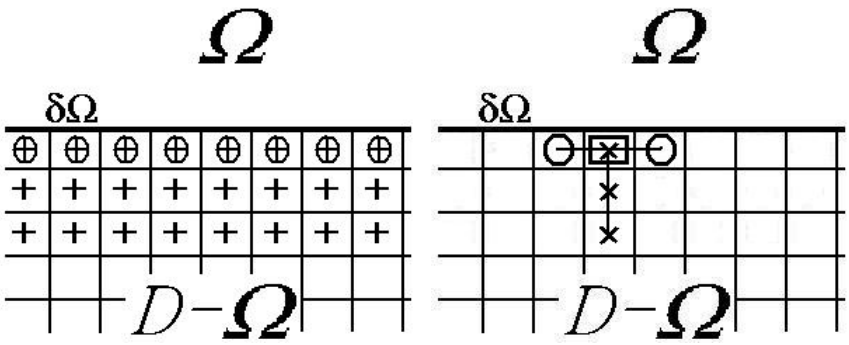

Figure 5: *An example of computing the boundary conditions on the bottom boundary.* Left: '+' marks where u and v are to be computed (three-pixel thickness) and 'o' marks where u_y, v_x and w (one-pixel thickness) are to be computed. Right: In computing u_y and v_x at the point on the boundary, use centered difference for v_x and a one-sided three-point formula for u_y.

2.4 The Anisotropic Diffusion $\nabla \cdot (g(|\nabla w|)\nabla w)$

The nested nature of the diffusion term in (5) demands a more careful treatment than ordinary diffusion of the form $\nu \Delta w$. We first note that:

$$\nabla \cdot (g(|\nabla w|)\nabla w) = \partial_x(g(|\nabla w|)w_x) + \partial_y(g(|\nabla w|)w_y) \tag{7}$$

The method used to discretize (7) works as follows:

1. Apply centered difference where possible and one-sided three-point formulae for boundaries to compute w_x and w_y at each grid point.
2. Compute $|\nabla w| = \sqrt{w_x^2 + w_y^2}$ and evaluate the predefined function $g(|\nabla w|)$ at each grid point.
3. Apply centered difference where possible and one-sided three-point formulae for boundaries to compute $\partial_x(g(|\nabla w|)w_x)$ and $\partial_y(g(|\nabla w|)w_y)$ at each grid point. Add to get the desired value at each grid point.

The predefined function used was the Perona-Malik function (6).

2.5 Treatment of Irregular Domains

Very few real applications of inpainting have rectangular Ω; Most Ω are highly irregular in shape, from scratches to fonts ([1],[2],[3]).
Suppose now that Ω is any arbitrary shape in D. Let $\bar{\Omega} \subset D$ be the smallest rectangular domain such that $\Omega \subset \bar{\Omega}$. With the formulation of the algorithm so far, one can inpaint on $\bar{\Omega}$, a larger region (See Figure 6). However, this will not efficiently use the information known in $\Omega \backslash \bar{\Omega}$; Since this algorithm only knows how to propagate the smoothness, ΔI, other information such as textures and complicated patterns will likely be lost if $\bar{\Omega}$ is large.
One method to overcome this problem is to solve (5) and the poisson equation in the irregular domain Ω. This method, however, faces difficulty in evolving the finite difference scheme

and implementing the boundary conditions.
The approach we took for this problem was as follows [5]: solve (5) using FTCS or FTUp and the poisson equation $\Delta I^n = w^n$ in the larger region $\bar{\Omega}$, then assign values into I for indicies in the region $\Omega \backslash \bar{\Omega}$ *at each time step*. In other words, we force the known values of I at each iteration wherever possible. Therefore, essentially, only the values in Ω vary after every iteration, just as desired. If a steady state solution is met, then by the continuity of the Navier-Stokes equations, the values of I in Ω should obey the Solution Criterion.

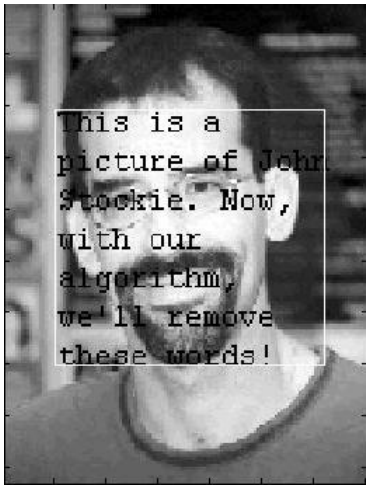

Figure 6: *An example of an Irregular domain.* Ω in this case are the black letters superimposed on the man's face. $\bar{\Omega}$ is the region inside the rectangle in white. The inpainting is performed in $\bar{\Omega}$. (image courtesy of John Stockie)

The advantages of this last method for irregular regions is clear. The implementation is extremely simple: only few extra lines of MATLAB codes are required to be modifed from the rectangular domain algorithm. Also, as will be shown later, the results produced from this simple approach is quite satisfactory.

2.6 Remarks on Accuracy, Stability and Continuity

The overall accuracy of this algorithm is $O(\Delta t, \Delta x)$ for FTUp and $O(\Delta t, \Delta x^2)$ for FTCS. The boundary condition implementation described in section 2.3, ensures that the second order accuracy is met at the boundaries. Generally in the fluid dynamics community, first order in space is considered too crude an approximating due to excessive diffusion. However, as we will see later, the numerical solutions to the inpainting problem using FTCS and FTUp appears nearly identical to the human eye. Note also that the restriction on Δt for stability is less tight for FTUp than for FTCS for linear advection-diffusion equations; this fact was experimentally shown to carry over for our inpainting algorithm.
Stability in general is currently a difficult issue. Bertozzi, et. al. states in [1]:

[5]None of the papers on inpainting that were studied contained hints on how to inpaint on irregular domains.

We expect that Navier-Stokes based inpainting may inherit some of the stability and uniqueness issues known for incompressible fluids, although the effect of anisotropic diffusion is not clear.

The non-linearity of the problem poses difficulty in analysing the stability condition analytically. Also, how the implementation for the irregular domains (forcing values at certain locations for every time step) affects the stability is difficult to predict. Needless to say, stability is a very important regarding the efficiency of the algorithm. Determining the maximum possible Δt allows few time step iterations to carry the solution to a steady state, hense speeding up the inpainting process. Numerical analysis regarding stability will be done later in Section 3.2.
Finally, as mentioned briefly earlier, the continuity of the Navier-Stokes equations guarantees several useful facts:

1. As developed in Section 1, the Solution Criterion is (nearly) satisfied for small ν.
2. The boundary condition of the poisson equation guarantees that I is continuous at $\partial\Omega$.
3. Specifying the tangential velocity vector from $\vec{v} = \nabla^{\perp} I$ guarantees the the direction of the isophotes are continuous at $\partial\Omega$. (See Figure 7)

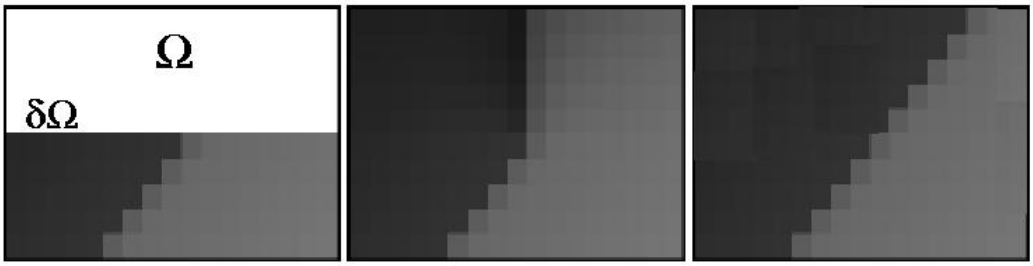

Figure 7: *The direction of isophotes on the boundary.* Left: Before inpainting. Center: After inpainting with an algorithm that does not satisfy continuous isophote direction on the boundary. Right: Inpainting with Navier-Stokes. Note how the isophote directions are continuous with the Navier-Stokes method.

3 Numerical Results

3.1 Examples

3.1.1 Rectangular Domain

A rectangular domain of size 20×20 was inpainted. Parameter used were: $\Delta t = 0.001$, $\nu = 1/\Delta t$, $K = 10^{-16}$. The inpainting process took a few seconds on a standard PC on MATLAB. See Figure 8.

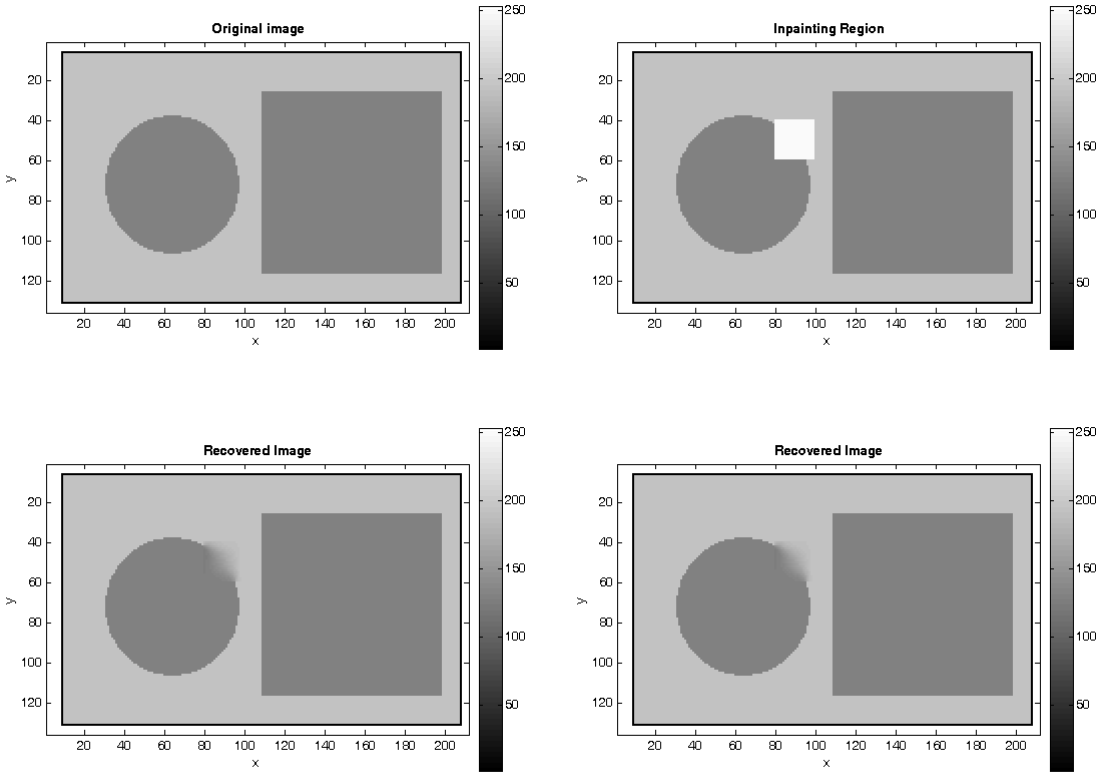

Figure 8: Top Left: Original image I. Top Right: Inpainting region shown as a white rectangle. Bottom Left: Inpainting result using the FTCS scheme. Bottom Right: Result using the FTUp scheme. Note how the two schemes produce nearly identical results.

3.1.2 Irregular Domain

For this artificial example, the inpainting was performed for a predefined Ω. It demonstrates the effect of using multiple small inpainting regions compared to one large region. Parameter used were: $\Delta t = 0.00001$, $\nu = 1/\Delta t$, $K = 10^{-16}$. The inpainting process took a few seconds on a standard PC on MATLAB. See Figures 9, 10, 11.
The next example is more practical. Parameter used were: $\Delta t = 0.00001$, $\nu = 2$, $K = 10^{-16}$. The inpainting process took a few seconds on a standard PC on MATLAB. See Figure 12.

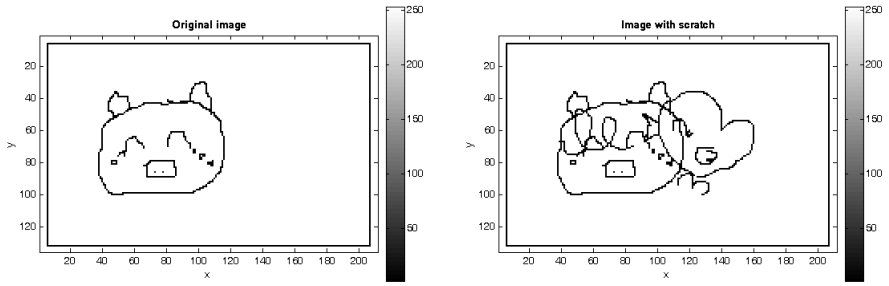

Figure 9: Left: Original image I. Right: The original image with graffiti (of a chicken). The goal is to recover the original image I.

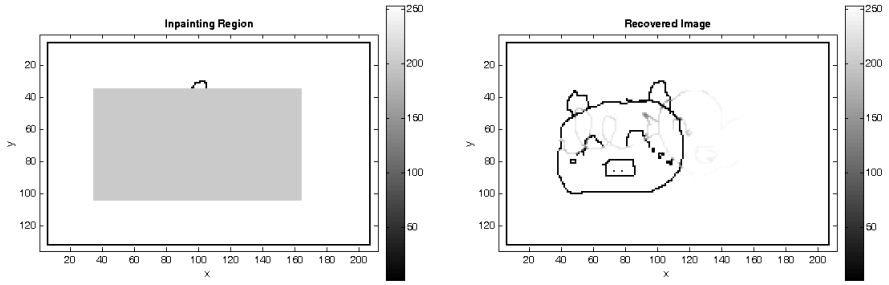

Figure 10: Left: Using a large inpainting domain. Right: Solution with FTUp using the large inpainting domain.

3.2 Attempts on Stability of Solutions

While testing with various sample images, we found that inpainting in a equally sized (rectangular) Ω and the same viscosity ν and Δt, produced stable solutions in some images and unstable solutions (graylevel blows up to $\pm\infty$) in others. It demonstrated that certain characteristics of I near $\partial\Omega$ has some sort of effect on the maximum stable choice for Δt. Recall that, for example, numerical solutions to the Navier-Stokes equations for the Driven-Cavity flow problem ([8]) using the FTCS scheme have stiffer restrictions on Δt as Re, the Reynolds number, increases. One can expect a similar phenomenon for inpainting using the Navier-Stokes equations.
The Reynolds number, for fluids, is defined to be,

$$Re = \frac{VL}{\nu},$$

where V is the characteristics velocity scale, L the length scale, and ν the viscosity. If the inpainting process were done in a fixed size (rectangular) Ω and ν, we anticipated that $|\nabla^\perp I|$ on $\partial\Omega$, which corresponds to $|\vec{v}|$ in fluids, has some effect on the maximum allowable Δt. Suppose we let $|\nabla^\perp I|_{ave}$ to denote the average value of $|\nabla^\perp I|$ along $\partial\Omega$ and $|\nabla^\perp I|_{max}$ to denote the maximum value of $|\nabla^\perp I|$ along $\partial\Omega$. Upon testing with the 'Lena' image at different pixel locations with Ω 30×30 pixels and $\nu = 2$, the result shown in Figure 13 was found between the ratio $|\nabla^\perp I|_{ave}/|\nabla^\perp I|_{max}$ and the maximum allowable Δt to attain a stable solution.

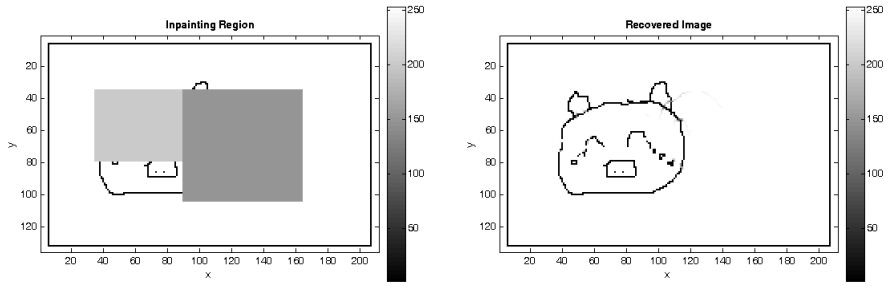

Figure 11: Left: Using two smaller inpainting domain. Right: Solution with FTUp using the two smaller inpainting domains. Note the better result with the smaller domains.

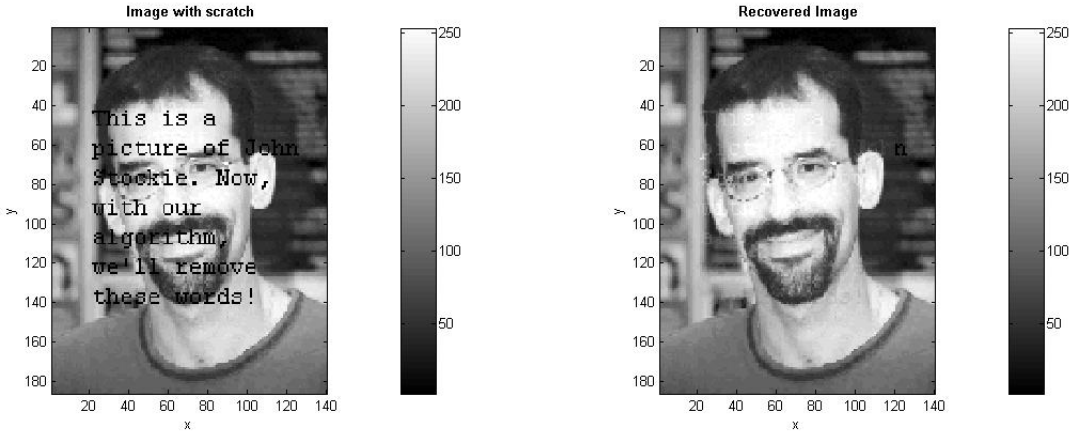

Figure 12: Left: The image with superimposed letters. Right: Inpainting solution with FTUp.

Knowledge about the choice of the maximum allowable Δt is at best, far from understood. Although Figure 13 suggests that images with larger $|\nabla^\perp I|_{ave}/|\nabla^\perp I|_{max}$ values have more related restriction on Δt, more analysis is necessary for its practical use, or even regarding its validity. For example, Bertozzi, et. al. in [1], on choosing the parameters, simply states:

> *Parameters for the algorithm have been chosen in such a way as to work for a wide range of examples:* $dt = 0.01$, $dx = dy = 1$, ...

4 Concluding Remarks

The image inpainting problem, and a method of solving it using techniques from CFD was introduced. This is a textbook example of interdisciplinary mathematics, exploiting techniques from a mature field in a relatively new field. Preliminary results show that the algorithm is satisfactory in practice, both in terms of the quality of the solution and computational efficiency. However, as mentioned earlier, stability of the algorithm is still an issue yet to be clarified.

Further minor extensions to this algorithm are possible. For example, the idea of inpainting

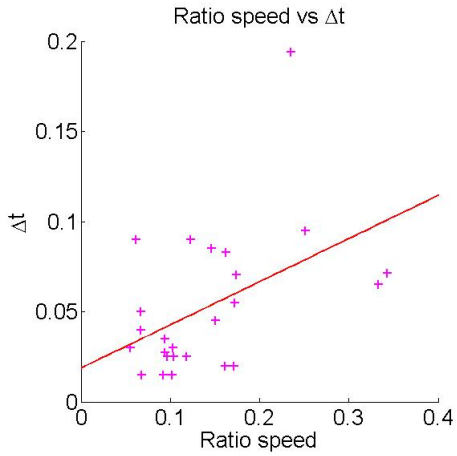

Figure 13: The horizontal and vertical axes represents $|\nabla^{\perp} I|_{ave}/|\nabla^{\perp} I|_{max}$ and the maximum allowable Δt, respectively. The crosses show the results from numerical tests. The line is the linear least squares fit of the data. The least squares fit was: $y = .1870677853 \times 10^{-1} + .239375186065265200x$.

on several smaller domains rather than one large domain for irregular regions may be further perfected. Actually implementing the algorithm on a irregular domain and not on the enclosing rectangle, $\bar{\Omega}$, may also be possible. Color images and video inpainting have already been successfully treated in [1].

5 Acknowledgments

The authors would like to thank Dr. John Stockie for his service in teaching us basic CFD techniques. Comments and suggestions on implementing the irregular domains and the Upwind methods from Dr. Adam Oberman were also appreciated.

References

[1] M. Bertalmio, A. L. Bertozzi, G. Sapiro, "Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting", Proceedings of the International Conference on Computer Vision and Pattern Recognition , IEEE, Dec. 2001, Kauai, HI, volume I, pp. I-355-I362

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image Inpainting", SIGGRAPH 2000, pages 417-424

[3] M. Oliveira, B. Bowen, R. McKenna, Yu-Sung Chang, "Fast Digital Image Inpainting", Proceedings of the International Conference on Visualization, Imaging and Image Processing (VIIP 2001), Marbella, Spain, September 3-5, 2001

[4] John F. Wendt, *Computational Fluid Dynamics: an introduction*, second edition, 1990, Springer-Verlag

[5] M. Black, G. Sapiro, D. Marimont, "Robust Anisotropic Diffusion", IEEE Transaction on Image Processing, Vol. 7, No. 3, March 1998

[6] P. Perona, J. Malik, "Scale-Space and Edge Detection Using Anisotropic Diffusion", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 7, July 1990

[7] T. Sziranyi, I. Kopilovic, B.P. Toth, "Anisotropic Diffusion as a Preprocessing Step for Efficient Image Compression", Proceedings of Fourteenth International Conference on Pattern Recognition, Volume: 2, pages 1565-1567, 16-20 Aug 1998

[8] C. Pozrikidis, *Introduction to Theoretical and Computational Fluid Dynamics*, Oct 1996, Oxford Press