J. KSIAM Vol.13, No.3, 225–236, 2009

FAST AND AUTOMATIC INPAINTING OF BINARY IMAGES USING A PHASE-FIELD MODEL

DARAE JEONG[1], YIBAO LI[1], HYUN GEUN LEE[1], AND JUNSEOK KIM[1†]

[1]DEPARTMENT OF MATHEMATICS, KOREA UNIVERSITY, SEOUL 136-701, REPUBLIC OF KOREA
E-mail address: cfdkim@korea.ac.kr

ABSTRACT. Image inpainting is the process of reconstructing lost or deteriorated parts of images using information from surrounding areas. We propose a computationally efficient and fast phase-field method which uses automatic switching parameter, adaptive time step, and automatic stopping of calculation. The algorithm is based on an energy functional. We demonstrate the performance of our new method and compare it with a previous method.

1. INTRODUCTION

Image inpainting[4, 1, 6] is the process of reconstructing lost or deteriorated parts of images using information from surrounding areas. Let $f(\mathbf{x})$, where $\mathbf{x} = (x, y)$, be a given image in a domain Ω. Let $c(\mathbf{x}, t)$ be a phase-field which is governed by the following modified Cahn-Hilliard (CH) equation[5]:

$$c_t = \Delta\mu + \lambda(\mathbf{x})(f(\mathbf{x}) - c), \tag{1.1}$$

$$\mu = F'(c) - \epsilon^2\Delta c, \tag{1.2}$$

where $F(c) = 0.25c^2(1-c)^2$. In the examples considered here, we use binary images in which most of the pixels are either exactly black or white. Eqs. (1.1) and (1.2) are the modified CH equation, due to the added fidelity term $\lambda(\mathbf{x})(f(\mathbf{x})-c)$ [2]. Image inpainting using phase-field methods is recently investigated by authors in [2, 3]. It is a good starting point for using partial differential equations in inpainting images, however, we found there are a couple of defects. First of all, switching parameter ϵ and stopping the calculation are done by trial and error. Furthermore, large time step Δt is more or less time step rescaling and it turns out that it is equivalent to using smaller time step than usual usage. In this paper, we propose a phase-field method which uses automatic varying ϵ, adaptive time step, and a stopping criterion based on energy functional.

Received by the editors June 26 2009; Accepted September 14 2009.
2000 *Mathematics Subject Classification.* 65M06, 65M55.
Key words and phrases. Binary images, Cahn-Hilliard equation, image inpainting.
† Corresponding author.

The outline of this paper is the following. In Sec. 2, the discrete equations for the governing equations are presented. In Sec. 3, we present computational examples. We propose a new automatic controlled algorithm in Sec. 4. Finally, in Sec. 5, conclusions are drawn.

2. DISCRETE EQUATIONS AND A NUMERICAL SOLUTION

In this section, we present fully discrete schemes for the CH equation in two dimensional space, i.e., $\Omega = (a, b) \times (c, d)$. Let N_x and N_y be positive even integers, $h = (b - a)/N_x$ be the uniform mesh size, and $\Omega_h = \{(x_i, y_j) : x_i = (i - 0.5)h,\ y_j = (j - 0.5)h,\ 1 \le i \le N_x,\ 1 \le j \le N_y\}$ be the set of cell-centers. Let c_{ij} and ν_{ij} be approximations of $c(x_i, y_j)$ and $\nu(x_i, y_j)$. Then, a semi-implicit time and centered difference space discretization of Eqs. (1.1) and (1.2) is

$$\begin{aligned} \frac{c_{ij}^{n+1} - c_{ij}^{n}}{\Delta t} &= \Delta_d \mu_{ij}^{n+\frac{1}{2}} + \lambda_{ij}(f_{ij} - c_{ij}^{n}), && (2.1)\\ \mu_{ij}^{n+\frac{1}{2}} &= \varphi(c_{ij}^{n+1}) - \frac{c^n}{4} - \epsilon^2 \Delta_d\, c_{ij}^{n+1}, && (2.2)\\ \text{where} \qquad & \varphi(c) = F'(c) + \frac{c}{4}. \end{aligned}$$

We can rewrite Eqs. (2.1) and (2.2) as follows:

$$\begin{aligned} \frac{c_{ij}^{n+1} - c_{ij}^{n}}{\Delta t} &= \Delta_d \nu_{ij}^{n+1} - \frac{1}{4}\Delta_d c_{ij}^{n} + \lambda_{ij}(f_{ij} - c_{ij}^{n}), && (2.3)\\ \nu_{ij}^{n+1} &= \varphi(c_{ij}^{n+1}) - \epsilon^2 \Delta_d c_{ij}^{n+1}. && (2.4) \end{aligned}$$

For completeness, the numerical solution using a nonlinear multigrid method is described. We use nonlinear Full Approximation Storage (FAS) multigrid method to solve the nonlinear discrete system (2.3) and (2.4) at the implicit time level. The nonlinearity is treated using one step of Newton's iteration and a pointwise Gauss-Seidel relaxation scheme is used as the smoother in the multigrid method. See the reference text [10] for additional details and backgrounds. The algorithm of the nonlinear multigrid method for solving the discrete CH system is :
First, let us rewrite Eqs. (2.3) and (2.4) as follows.

$$NSO(c^{n+1}, \nu^{n+1}) = (\phi^n, \ \ ^n),$$

where

$$NSO(c^{n+1}, \nu^{n+1}) = \left(\frac{c^{n+1}}{\Delta t} - \Delta_d \nu^{n+1}, \nu^{n+1} - \varphi(c^{n+1}) + \epsilon^2 \Delta_d c^{n+1} \right)$$

and the source term is

$$(\phi^n, \ \ ^n) = \left(\frac{c^n}{\Delta t} + \lambda(f - c^n) - \frac{1}{4}\Delta_d c^n,\ 0 \right).$$

In the following description of one FAS cycle, we assume a sequence of grids Ω_k (Ω_{k-1} is coarser than Ω_k by factor 2). Given the number β of pre- and post- smoothing relaxation sweeps, an iteration step for the nonlinear multigrid method using the V-cycle is formally

written as follows [10]:

FAS multigrid cycle

$$\{c_k^{m+1}, \nu_k^{m+1}\} = FAScycle(k, c_k^m, \nu_k^m, NSO_k, \phi_k^n, \ {}^n_k, \beta).$$

That is, $\{c_k^m, \nu_k^m\}$ and $\{c_k^{m+1}, \nu_k^{m+1}\}$ are the approximations of $c^{n+1}(x_i, y_j)$ and $\nu^{n+1}(x_i, y_j)$ before and after a FAScycle. Now, define the FAScycle.

1) Presmoothing

$$\{\bar{c}_k^m, \bar{\nu}_k^m\} = SMOOTH^\beta(c_k^m, \nu_k^m, NSO_k, \phi_k^n, \ {}^n_k),$$

which means performing β smoothing steps with the initial approximations c_k^m, ν_k^m, source terms ϕ_k^n, ${}^n_k$, and $SMOOTH$ relaxation operator to get the approximations $\bar{c}_k^m, \bar{\nu}_k^m$. One $SMOOTH$ relaxation operator step consists of solving the system (**??**) and (**??**) given below by 2×2 matrix inversion for each i and j. Here, we derive the smoothing operator in two dimensions. Rewriting Eq. (2.3), we get

$$\frac{c_{ij}^{n+1}}{\Delta t} + \frac{4\nu_{ij}^{n+1}}{h^2} = \phi_{ij}^n + \frac{\nu_{i+1,j}^{n+1} + \nu_{i-1,j}^{n+1} + \nu_{i,j+1}^{n+1} + \nu_{i,j-1}^{n+1}}{h^2}. \tag{2.5}$$

Since $\varphi(c_{ij}^{n+1})$ is nonlinear with respect to c_{ij}^{n+1}, we linearize $\varphi(c_{ij}^{n+1})$ at c_{ij}^m, i.e.,

$$\varphi(c_{ij}^{n+1}) \approx \varphi(c_{ij}^m) + \frac{d\varphi(c_{ij}^m)}{dc}(c_{ij}^{n+1} - c_{ij}^m).$$

After substitution of this into (2.4), we get

$$\begin{aligned} -\left(\frac{d\varphi(c_{ij}^m)}{dc} + \frac{4\epsilon^2}{h^2}\right) c_{ij}^{n+1} + \nu_{ij}^{n+1} \quad &= \quad {}^n_{ij} + \varphi(c_{ij}^m) - \frac{d\varphi(c_{ij}^m)}{dc} c_{ij}^m \\ &\quad - \frac{\epsilon^2}{h^2}(c_{i+1,j}^{n+1} + c_{i-1,j}^{n+1} + c_{i,j+1}^{n+1} + c_{i,j-1}^{n+1}). \end{aligned} \tag{2.6}$$

Next, we replace c_{kl}^{n+1} and ν_{kl}^{n+1} in the Eqs. (2.5) and (2.6) with $\bar{c}_{kl}^m$ and $\bar{\nu}_{kl}^m$ if $k \le i$ and $l \le j$, otherwise with c_{kl}^m and ν_{kl}^m, i.e.,

$$\frac{\bar{c}_{ij}^m}{\Delta t} + \frac{4\bar{\nu}_{ij}^m}{h^2} = \phi_{ij}^n + \frac{\nu_{i+1,j}^m + \bar{\nu}_{i-1,j}^m + \nu_{i,j+1}^m + \bar{\nu}_{i,j-1}^m}{h^2},$$

$$\begin{aligned} -\left(\frac{d\varphi(c_{ij}^m)}{dc} + \frac{4\epsilon^2}{h^2}\right) \bar{c}_{ij}^m + \bar{\nu}_{ij}^m \quad &= \quad {}^n_{ij} + \varphi(c_{ij}^m) - \frac{d\varphi(c_{ij}^m)}{dc} c_{ij}^m \\ &\quad - \frac{\epsilon^2}{h^2}(c_{i+1,j}^m + \bar{c}_{i-1,j}^m + c_{i,j+1}^m + \bar{c}_{i,j-1}^m). \end{aligned}$$

2) Compute the defect

$$(\bar{d_1}^m{}_k, \bar{d_2}^m{}_k) = (\phi_k^n, \ {}^n_k) - NSO_k(\bar{c}_k^m, \bar{\nu}_k^m).$$

3) Restrict the defect and $\{\bar{c}_k^m, \bar{\nu}_k^m\}$

$$(\bar{d_1}^m{}_{k-1}, \bar{d_2}^m{}_{k-1}) = I_k^{k-1}(\bar{d_1}^m{}_k, \bar{d_2}^m{}_k), \ (\bar{c}_{k-1}^m, \bar{\nu}_{k-1}^m) = I_k^{k-1}(\bar{c}_k^m, \bar{\nu}_k^m).$$

The restriction operator I_k^{k-1} maps k-level functions to $(k-1)$-level functions.

$$\begin{aligned} d_{k-1}(x_i, y_j) = I_k^{k-1} d_k(x_i, y_j) &= \frac{1}{4}[d_k(x_{i-\frac{1}{2}}, y_{j-\frac{1}{2}}) + d_k(x_{i-\frac{1}{2}}, y_{j+\frac{1}{2}}) \\ &\quad + d_k(x_{i+\frac{1}{2}}, y_{j-\frac{1}{2}}) + d_k(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}})]. \end{aligned}$$

4) Compute the right-hand side

$$(\phi_{k-1}^n, \ \ {}_{k-1}^n) = (\bar{d_1^m}_{k-1}, \bar{d_2^m}_{k-1}) + NSO_{k-1}(\bar{c}_{k-1}^m, \bar{\nu}_{k-1}^m).$$

5) Compute an approximate solution $\{\hat{c}_{k-1}^m, \hat{\nu}_{k-1}^m\}$ of the coarse grid equation on Ω_{k-1}, i.e.

$$NSO_{k-1}(c_{k-1}^m, \nu_{k-1}^m) = (\phi_{k-1}^n, \ \ {}_{k-1}^n). \tag{2.7}$$

If $k = 1$, we apply the smoothing procedure in (1) to obtain the approximate solution. If $k > 1$, we solve (2.7) by performing a FAS k-grid cycle using $\{\bar{c}_{k-1}^m, \bar{\nu}_{k-1}^m\}$ as an initial approximation:

$$\{\hat{c}_{k-1}^m, \hat{\nu}_{k-1}^m\} = \text{FAScycle}(k-1, \bar{c}_{k-1}^m, \bar{\nu}_{k-1}^m, NSO_{k-1}, \phi_{k-1}^n, \ \ {}_{k-1}^n, \beta).$$

6) Compute the coarse grid correction (CGC):

$$\hat{v}_{1k-1}^m = \hat{c}_{k-1}^m - \bar{c}_{k-1}^m, \ \hat{v}_{2k-1}^m = \hat{\nu}_{k-1}^m - \bar{\nu}_{k-1}^m.$$

7) Interpolate the correction: $\hat{v}_{1k}^m = I_{k-1}^k \hat{v}_{1k-1}^m, \ \hat{v}_{2k}^m = I_{k-1}^k \hat{v}_{2k-1}^m$.

Here, the coarse values are simply transferred to the four nearby fine grid points, i.e. $v_k(x_i, y_j) = I_{k-1}^k v_{k-1}(x_i, y_j) = v_{k-1}(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}})$ for i and j odd-numbered integers.

8) Compute the corrected approximation on Ω_k

$$c_k^{m,\text{ after } CGC} = \bar{c}_k^m + \hat{v_1}_k^m, \ \nu_k^{m,\text{ after } CGC} = \bar{\nu}_k^m + \hat{v_2}_k^m.$$

9) Postsmoothing

$$\{c_k^{m+1}, \nu_k^{m+1}\} = SMOOTH^{\beta}(c_k^{m,\text{ after } CGC}, \nu_k^{m,\text{ after } CGC}, NSO_k, \phi_k^n, \ \ {}_k^n).$$

This completes the description of a nonlinear FAScycle for the discrete modified CH equation. Let us define a maximum norm

$$\|c^n\|_\infty = \max_{\substack{1 \le i \le N_x \\ 1 \le j \le N_y}} |c_{ij}^n|.$$

3. Computational examples

In this section, we will compare the numerical scheme of the previous Bertozzi's paper [2] with our scheme. First we refer to the discrete Eq. (9) in the paper [2]

$$\begin{aligned} \frac{u^{n+1} - u^n}{\Delta t} &+ \varepsilon {\Delta^2}_d u^{n+1} - C_1 \Delta_d u^{n+1} + C_2 u^{n+1} \\ &= \Delta_d \left(\frac{1}{\varepsilon} \mathrm{W}'(u^n) \right) + \lambda(\mathbf{x})(f(x) - u^n) - C_1 \Delta_d u^n + C_2 u^n, \end{aligned} \tag{3.1}$$

where $W(u) = u^2(1-u)^2$ and the constants C_1 and C_2 are large enough so that the equation is convex for the range of u in the simulation. Next, we rewrite Eq. (3.1) as follows:

$$\frac{u^{n+1}-u^n}{\frac{4\Delta t}{\varepsilon(C_2\Delta t+1)}} = \Delta_d\left(\frac{1}{4}\mathrm{W}'(u^n) - \frac{\varepsilon^2}{4}\Delta_d u^{n+1} + \frac{\varepsilon}{4}C_1(u^{n+1}-u^n)\right) + \frac{\varepsilon}{4}\lambda(\mathbf{x})(f(\mathbf{x})-u^n).$$

Table 1 shows that two schemes are equivalent.

TABLE 1. Equivalent forms of two schemes.

Bertozzi's numerical scheme
$\frac{u^{n+1}-u^n}{\frac{4\Delta t}{\varepsilon(C_2\Delta t+1)}} = \Delta_d(\frac{1}{4}\mathrm{W}'(u^n) - \frac{\varepsilon^2}{4}\Delta_d u^{n+1} + \frac{\varepsilon}{4}C_1(u^{n+1}-u^n)) + \frac{\varepsilon}{4}\lambda(\mathbf{x})(f(\mathbf{x})-u^n)$
Our numerical scheme
$\frac{c^{n+1}-c^n}{\Delta t} = \Delta_d\left(F'(c^n) - \epsilon^2\Delta_d c^{n+1} + \frac{1}{4}(c^{n+1}-c^n)\right) + \lambda(\mathbf{x})(f(\mathbf{x})-c^n)$

We perform two test problems such as inpaintings of a double stripe and of a cross to show that two schemes are equivalent.

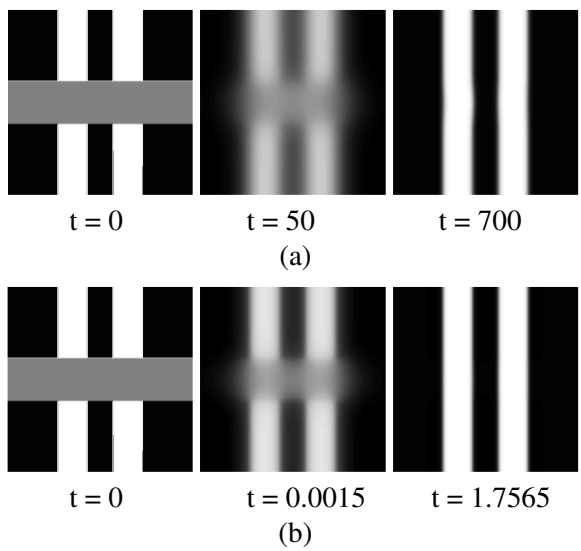

FIGURE 1. (a) Bertozzi's result. (b) Our result. Left column is initial data, middle column is results at iteration 50, and right column is results at iteration 700.

3.1. **Inpainting of a double stripe.** In this test problem, the computational domain $\Omega = (0, 1.28) \times (0, 1.28)$ and 128×128 mesh size are taken. The initial configurations are shown in the first column in Fig. 1. In the first and second rows, figures are results using the previous

scheme and our proposed scheme, respectively. The gray region in the initial configuration denotes the inpainting region. In the previous scheme, we start calculations with a large $\epsilon = 0.8$ value, then switch its value to $\epsilon = 0.01$ after 50 iterations, and stop the calculation at 700 iterations. We repeat the same calculations with equivalent values which are summarized in Table 2. The prime notations of the parameters are from previous method and right arrow implies changing values when switching happens. By comparing two results, we see that our scheme is equivalent to the previous Bertozzi's scheme.

TABLE 2. Equivalent parameter values of two schemes.

Previous	Value	Current	Value
$\Delta t'$	1.0	$\Delta t = \frac{4\Delta t'}{\epsilon'(C_2'\Delta t' + 1)}$	$0.00003 \rightarrow 0.0027$
λ'	50000	$\lambda = \frac{\epsilon'}{4}\lambda'$	$10000 \rightarrow 125$
ϵ'	$0.8 \rightarrow 0.01$	$\epsilon = \frac{\epsilon'}{2}$	$0.4 \rightarrow 0.005$

3.2. **Inpainting of a cross.** For the second test problem, the initial configuration is a cross with an inpainting region as shown in the first column in Fig. 2. The computational domain $\Omega = (0, 1.28) \times (0, 1.28)$ and 128×128 mesh size are taken. In the first and second rows, figures are results using the previous scheme and our proposed scheme, respectively. In the previous scheme, we start calculations with a large $\epsilon = 0.8$ value, switch its value to $\epsilon = 0.01$ after 300 iterations, and stop the calculation at 1000 iterations. We repeat the same calculations with equivalent values which are summarized in Table 3. By comparing two results, we see that our scheme is equivalent to the previous Bertozzi's scheme.

TABLE 3. Equivalent parameter values of two schemes.

Previous	Value	Current	Value
$\Delta t'$	1.0	$\Delta t = \frac{4\Delta t'}{\epsilon'(C_2'\Delta t' + 1)}$	$0.000017 \rightarrow 0.0013$
λ'	100000	$\lambda = \frac{\epsilon'}{4}\lambda'$	$20000 \rightarrow 250$
ϵ'	$0.8 \rightarrow 0.01$	$\epsilon = \frac{\epsilon'}{2}$	$0.4 \rightarrow 0.005$

From these two test problems, we can conclude that two schemes are equivalent. However, in the previous algorithm, when to switch the parameters and when to stop the calculation are from trial and error. Therefore, it is our main purpose to propose an automatic switching and stopping algorithm based on an energy functional.

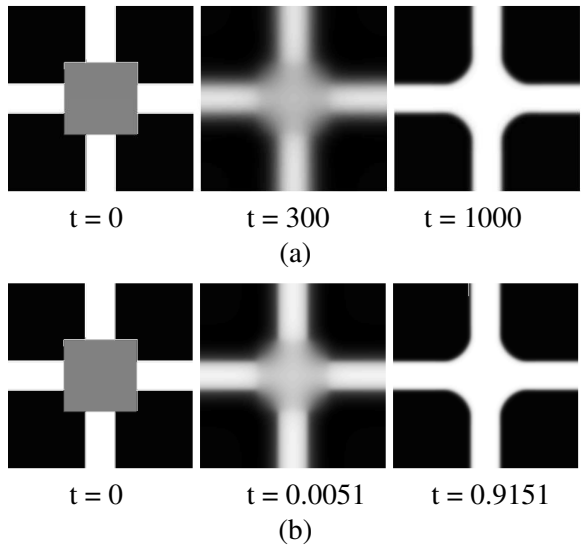

FIGURE 2. (a) Bertozzi's result. (b) Our result. Left column is initial data, middle column is results at iteration 300, and right column is results at iteration 1000.

4. PROPOSED ALGORITHM FOR AUTOMATIC CONTROL

In this section, we propose an automatic switching and stopping algorithm based on an energy functional. Let us reconsider the first test problem which is the inpainting of a double stripe. Fig. 3 shows the temporal evolution of contour plots of the phase-field. Around the switching time, we can observe the phase separation which means that the inpainting region separates into white and dark regions. Then we switched the ϵ parameter. Next, let us take a look at the time evolution of the energy functional. In Fig. 4, the energy is increased at the initial stage and it is decreased. Similar phenomena in the second test problem which is the inpainting of a cross are observed. See the Figs. 5 and 6. Therefore, it is natural to monitor the energy functional for switching the parameter and stopping the calculation.

4.1. **Inpainting of damaged images.** Fig. 7(a) and (c) show the initial images of damaged double stripes and cross and Fig. 7(b) and (d) show the results with our proposed automatic algorithm to a double stripe and a cross inpainting problems. In the case of double stipes (see Fig. 7(a) and (b)), it only requires 16 iterations to recover the damaged images. Also in the other case (cross image, see Fig. 7(c) and (d)) we obtain the recovered image after 15 iterations. We use $\Delta t = 1/128,\ \epsilon = 0.038424, \lambda = 3/\Delta t$ and when $diff$,the difference of energies of c^{n+1} and c^n, is smaller than $tol1(= 0.08)$ is equal to 3 times, that is, at the number of iteration is 3 in both cases, we switch the parameter as $\Delta t' = 2.0\Delta t,\ \epsilon' = 0.0875\epsilon,\ \lambda' = 1.8/\Delta t'$. When $diff$ is smaller than prescribed tolerance, $tol2\ (= 1.0E-6)$, we stop this algorithm.

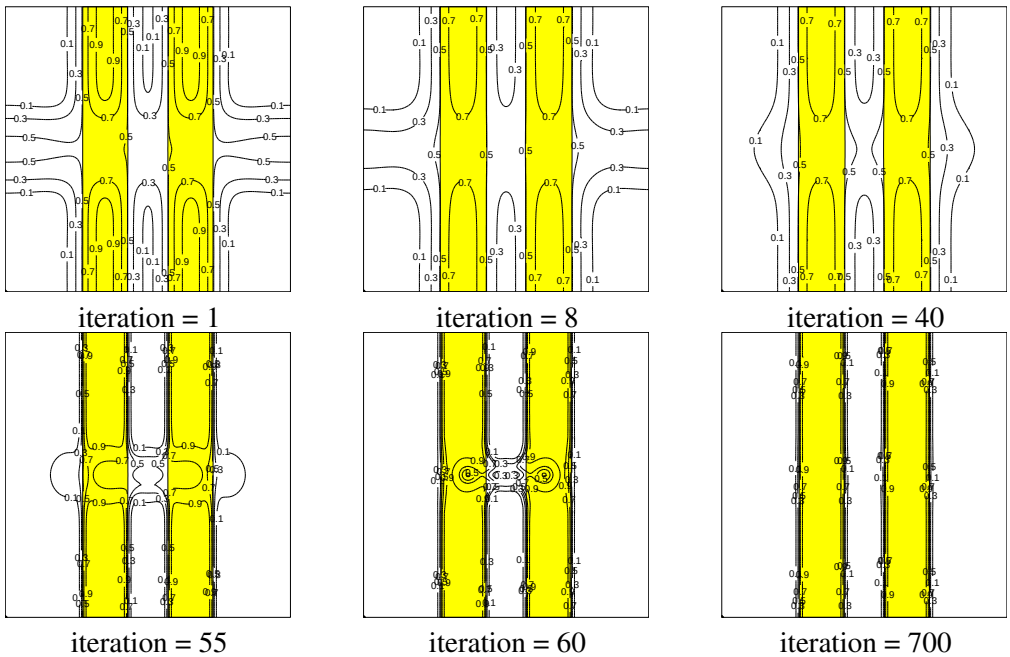

FIGURE 3. Temporal evolution of contour plots of the phase-field for the double stripe inpainting.

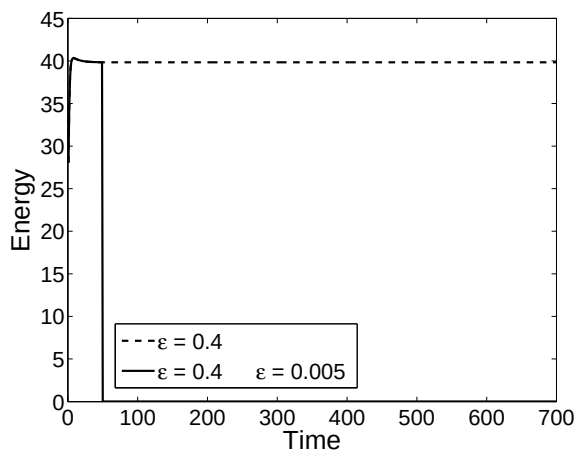

FIGURE 4. Temporal evolution of contour plots of the energy functional for the double stripe inpainting.

4.2. **Inpainting of obscured text.** In order to recover obscured text (see Fig. 8(a)), we use our inpainting algorithm. Fig. 8(a) shows the initial image which is obscured text by lines

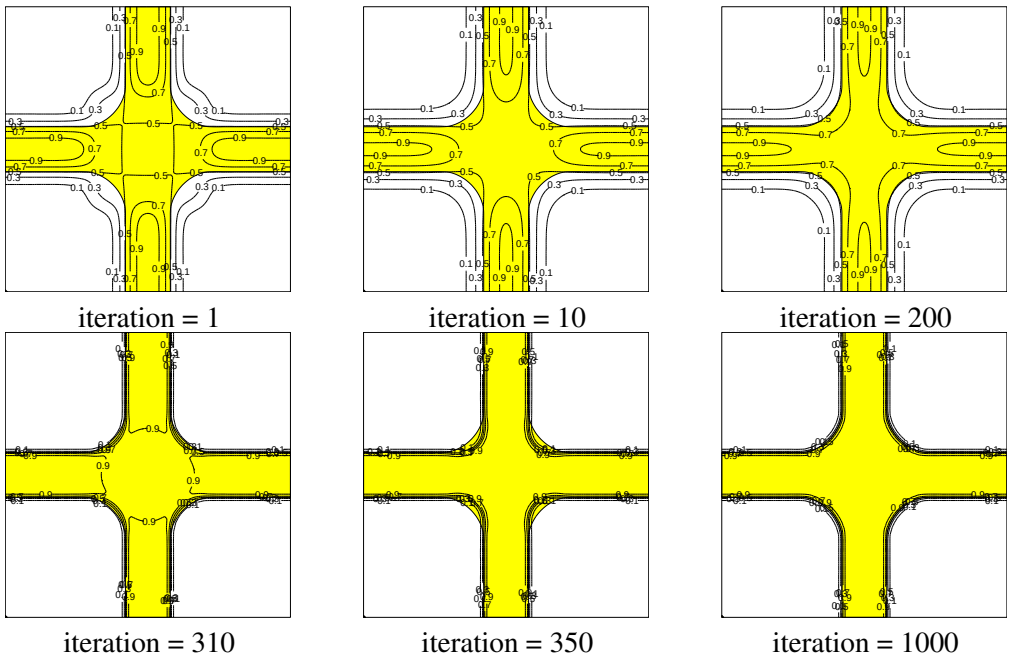

FIGURE 5. Temporal evolution of contour plots of the phase-field for the cross inpainting.

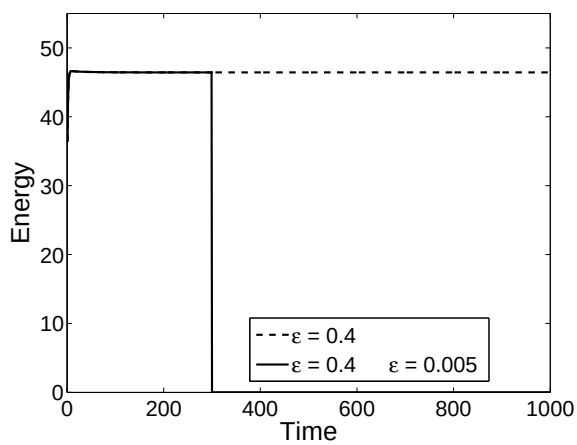

FIGURE 6. Temporal evolution of contour plots of the energy functional for the cross inpainting.

and Fig. 8(b) shows the recover result image. We use the parameter as $\Delta t = 1/128$, $\epsilon = 0.038424, \lambda = 3/\Delta t$ and $diff$ is smaller than $tol1$ $(= 0.08)$ is equal to 3 times, that is, at the number of iteration is 4 in this case, we switch the parameter as $\Delta t' = 1.8\Delta t$, $\epsilon' =$

The proposed automatic switching and stopping algorithm is as follows.
Algorithm:
Given a maximum iteration number N, tolerances $tol1$ and $tol2$
• Set $k = 1$, $flag = 0$.
• While $(k \leq N)$ do *Steps 1-4*

Step 1 Compute c^{n+1} from c^n by solving Eqs. (2.3) and (2.4).

Step 2 Check the difference of energies of c^n and c^{n+1}

$diff = |\mathcal{E}(c^n) - \mathcal{E}(c^{n+1})|$

If $(flag < 3$ and $diff < tol1)$

$flag = flag + 1$

End

Step 3 Switch parameters and reset data

If $(flag = 3)$

If $(c^{n+1} > 0.5)$

$c^{n+1} = 1$

Else

$c^{n+1} = 0$

End

$\Delta t = 2\Delta t$

do *Step 1* twice

$\epsilon = 0.0875\epsilon$

$\lambda = 1.8/\Delta t$

End

Step 4 Stop loop

If $(diff < tol2$ and $flag = 3)$

Stop loop

End

End

0.0875ϵ, $\lambda' = 1.8/\Delta t'$. Our automatic switching method of the modified CH equation is faster than the previous model.

5. Conclusion

We have shown that our automatic switching algorithm achieves faster inpainting of binary images than the previous trial and error algorithm. Therefore, inpainting region is reconstructed more efficiently and faster than previous method. The developed automatic algorithm can be applied to calculating option pricing such as the Black-Scholes equations accurately and efficiently.

(a) Initial (b) 16 iterations (c) Initial (d) 15 iterations

FIGURE 7. Recovery of damaged images. The computational domain is $\Omega = (0, 1.28) \times (0, 1.28)$ and mesh size is 128×128.

(a) Initial (b) 47 iterations

FIGURE 8. Recovery of damaged text. The computational domain is $\Omega = (0, 2.56) \times (0, 1.28)$ and mesh size is 256×128.

ACKNOWLEDGMENTS

This work was supported by the research fund (R0600842) of Seoul Research and Business Development Program.

REFERENCES

[1] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, *Image inpainting*, Proc. of SIGGRAPH 2000, New Orleans, USA 2000.

[2] A. Bertozzi, S. Esedoglu, and A. Gillette, *Inpainting of binary images using the Cahn-Hilliard equation*, IEEE Trans. Image Proc., **16** (2007), 285-291.

[3] A. Bertozzi, S. Esedoglu, and A. Gillette, *Analysis of a two-scale Cahn-Hilliard model for image inpainting*, Multiscale Modeling and Simulation, **6** (2007), 913-936.

[4] M. Burger, L. He, and C. Schöenlieb, *Cahn-Hilliard inpainting and a generalization for grayvalue images*, UCLA CAM report 08-41, 2008.

[5] J. W. Cahn and J. E. Hilliard, *Free energy of a nonuniform system. I. interfacial free energy*, J. Chem. Phys., **28** (1958), 258-267.

[6] T. F. Chan and J. Shen, *Mathematical models for local non-texture inpaintings*, SIAM J. Appl. Math., **62** (2001), 1019-1043.

[7] D. J. Eyre, http://www.math.utah.edu/∼eyre/research/methods/stable.ps.

[8] D. J. Eyre, in *Computational and mathematical models of microstructural evolution*, The Material Research Society, Warrendale, PA, (1998), 39-46.

[9] E. V. L. Mello and O. T. S. Filho, *Numerical study of the Cahn-Hilliard equation in one, two and three dimensions*, Physica A, **347** (2005), 429-443.
[10] U. Trottenberg, C. Oosterlee, and A. Schüller, *MULTIGRID*, Academic press, 2001.