Math 6590 Project V: Implementation and comparison of non-convex and convex Chan-Vese segmentation models

Name: Li Dong
RIN: 661243168

May 4, 2016

Abstract

This report implements and compares two popular image segmentation algorithms - the Chan-Vese algorithm and its convex version. We first briefly describe the algorithm and then show the discretization and itaration strategy. For the sake of fair comparison, gradient flow is used for both algorithms. Also, all parameters are kept the same for both algorithms in the image test case. In the end, we can draw some conclusions about the performance of the two algorithms based on their convergence and image segmentation results.

1 Problems

1. Describe the Chan-Vese segmentation model and the numerical method for image segmentation. Implement this model and test the attached image(use the clean image and the noisy image).
2. Describe the convex version of Chan-Vese segmentation model and the numerical method for image segmentation. Implement this model, test the attached image(use the clean image and the noisy image) and compare it with the original Chan-Vese model.

2 Methodology

2.1 Chan-Vese segmentation model

Let us first introduce the Chan-Vese model,

$$\begin{aligned}\min_{c_1,c_2,C} F(c_1,c_2,C) =& \mu \cdot Length(C) + \nu \cdot Area(inside(C)) + \\ & \lambda_1 \int_{inside(C)} |u_0 - c_1|^2 dxdy + \lambda_2 \int_{outside(C)} |u_0 - c_2|^2 dxdy,\end{aligned} \tag{1}$$

where $\mu \geq 0$, $\nu \geq 0$, $\lambda_1, \lambda_2 > 0$ are fixed parameters. Since we are minimizing the length of the segmentation curves, the area of the segmentation curves and matching the contents inside and outside the segmentation curves, it is intuitive that the minimizer of Eq.(1) is simply our segmentation curves.

The Chan-Vese segmentation model [1] is a successful approximation to the Mumford-Shah functional [2]. The Mumford-Shah functional is

$$\min_{u,C} F(u,C) = \mu \cdot Length(C) + \lambda \int_\Omega |u_0 - u|^2 d\Omega + \int_{\Omega \setminus C} \nabla u^2 d\Omega, \tag{2}$$

where u_0 is a given image, μ and λ are positive parameters. The solution image u obtained by minimizing this functional is formed by smooth regions with sharp boundaries, denoted by C. If we let $\nu = 0$ and assume u is a constant c_1 inside the boundaries C and a constant c_2 outside the boundaries C in Eq.(1) and omit the square derivative term in Eq.(2).

Due to the friendly behaviour in topological changes, level set method is the natural choice for image segmentation. Now let us represent the Chan-Vese model with level set method.

$$\begin{aligned}\min_{c_1,c_2,\phi} F(c_1,c_2,\phi) =&\mu\int_\Omega \delta(\phi)_\epsilon|\nabla\phi|dxdy+\nu\int_\Omega H(\phi)_\epsilon dxdy+\\ &\lambda_1\int_\Omega (u_0-c_1)^2H(\phi)_\epsilon dxdy+\lambda_2\int_\Omega (u_0-c_2)^2(1-H(\phi)_\epsilon)dxdy,\end{aligned} \tag{3}$$

where ϕ is a signed distance function and takes positive values inside C, negative values outside C and zeros on C, H_ϵ is a smoothed Heaviside function and δ_ϵ is a smoothed Delta function which is the derivative of the Heaviside function. Specifically,

$$H(x)_\epsilon = \begin{cases} 1, & x>\epsilon \\ 0, & x<-\epsilon \\ \frac{1}{2}(1+\frac{2}{\pi}arctan(\frac{x}{\epsilon})), & |x|<\epsilon \end{cases}$$

Now let us state the algorithm.

1. Fix ϕ, solve c_1 and c_2.

$$\frac{\delta F}{\delta c_1}=0 \quad\Rightarrow\quad c_1=\frac{\int_\Omega u_0H(\phi)d\Omega}{\int_\Omega H(\phi)d\Omega}, \tag{4}$$

$$\frac{\delta F}{\delta c_2}=0 \quad\Rightarrow\quad c_2=\frac{\int_\Omega u_0(1-H(\phi))d\Omega}{\int_\Omega (1-H(\phi))d\Omega}. \tag{5}$$

2. Fix c_1 and c_2, solve ϕ. $\frac{\delta F(c_1,c_2,\phi)}{\delta\phi}=0$ gives the following gradient flow.

$$\frac{\delta\phi}{\delta t} = \delta(\phi)(\mu div(\frac{\nabla\phi}{|\nabla\phi|})-\lambda_1(u_0-c_1)^2+\lambda_2(u_0-c_2)^2-\nu), \tag{6}$$

$$\phi(0,x,y) = \phi_0(x,y), \tag{7}$$

$$\frac{\delta(\phi)}{|\nabla\phi|}\frac{\delta\phi}{\delta\vec{n}} = 0, \quad on \quad \delta\Omega. \tag{8}$$

If we define Δ_+^m, Δ_-^m and Δ_c^m to be the forward, backward and central difference and $m=\{x,y\}$, we can have the following discretization for the previous gradient flow.

$$\begin{aligned}\phi_{ij}^{n+1} =&\phi_{ij}^n+dt\cdot\delta(\phi_{ij}^n)(\mu\Delta_-^x(\frac{\Delta_+^x\phi_{ij}^n}{\sqrt{(\Delta_+^x\phi_{ij}^n)^2+(\Delta_c^y\phi_{ij}^n)^2+\beta^2}})+\mu\Delta_-^y(\frac{\Delta_+^y\phi_{ij}^n}{\sqrt{(\Delta_c^x\phi_{ij}^n)^2+(\Delta_+^y\phi_{ij}^n)^2+\beta^2}})-\\ &\nu-\lambda_1(u_{0,ij}-c_1)^2+\lambda_2(u_{0,ij}-c_2)^2).\end{aligned} \tag{9}$$

2.2 Convex version of Chan-Vese segmentation model

Since the Chan-Vese model in the previous section is non-convex, a convex version of the Chan-Vese model was later proposed [3]. We can observe that we can have the same solution as the gradient flow in Eq.(6) by using the following gradient flow,

$$\frac{\delta\phi}{\delta t} = div(\frac{\nabla u}{|\nabla\phi|})-\lambda(c_1-u_0)^2-\lambda(c_2-u_0)^2. \tag{10}$$

Furthermore, we can observe that the energy functional to the above gradient flow is as follows.

$$\min_{0\leq\phi\geq 1} E(\phi) = \int_\Omega |\nabla\phi| d\Omega + \lambda \int_\Omega ((c_1 - u_0)^2 - (c_2 - u_0)^2)\phi d\Omega. \tag{11}$$

Although the augmented Lagrangian can be used to solve the above energy functional, we choose to use the same strategy as the non-convex Chan-Vese model, so that we can compare the results fairly.

Now we state the algorithm as follows.

1. Solve ϕ using the explicit finite difference strategy.

$$\begin{aligned}\phi_{ij}^{n+1} =& \phi_{ij}^n + dt(\Delta_-^x(\frac{\Delta_+^x \phi_{ij}^n}{\sqrt{(\Delta_+^x\phi_{ij}^n)^2 + (\Delta_c^y\phi_{ij}^n)^2 + \beta^2}}) + \Delta_-^y(\frac{\Delta_+^y \phi_{ij}^n}{\sqrt{(\Delta_c^x\phi_{ij}^n)^2 + (\Delta_+^y\phi_{ij}^n)^2 + \beta^2}}) - \\ & - \lambda(u_{0,ij} - c_1)^2 + \lambda(u_{0,ij} - c_2)^2).\end{aligned} \tag{12}$$

   We note that this is very similar to Eq.(9).

2. Normalize ϕ by $\phi \leftarrow max\{min\{\phi, 1\}, 0\}$.

3. $c_1 = \frac{\int_\Sigma u_0}{|\Sigma|}$, $c_2 = \frac{\int_{\Sigma^c} u_0}{|\Sigma^c|}$, where $\Sigma = \{\phi > \eta\}$, $0 < \eta < 1$ and $|\Sigma|$ is the area of Σ. We initialize $c_1 = 0$,$c_2 = 0$. We note that the value of the thresholding parameter η does not affect the solution too much.

2.3 Remark

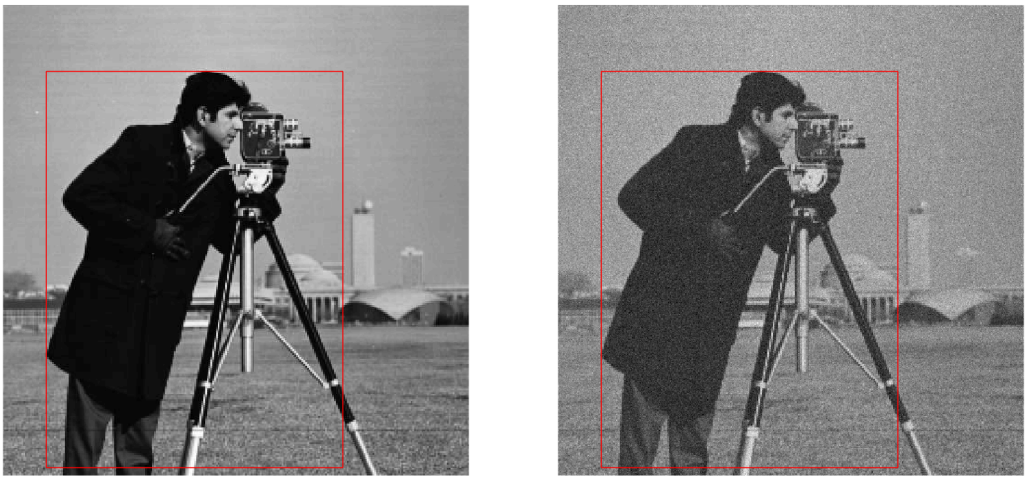
Since we want fair comparison, we have the following parameters setup for both models and for both test cases. For the non-convex Chan-Vese model, $\lambda_1 = \lambda_2 = 0.1, \mu = 1, \nu = 0, \epsilon = 10$. For the convex Chan-Vese model, $\lambda = 0.1, \eta = 0.2$. For both models, we use $\beta = 1 \times 10^{-8}, dt = 0.1$ and the algorithm terminates when either $\frac{||\phi^n - \phi^{n-1}||}{||\phi^{n-1}||} < 1 \times 10^{-10}$ or it reaches 300 iterations. Moreover, reinitialization is optional for the Chan-Vese model. In this study, no reinitialization is used.

3 Results

In this section, we compare the performance of the non-convex and convex Chan-Vese models by applying the models to the same image without and with noise. Unlike the previous active contour models, such as the snake model and the geodesic model, Chan-Vese model is not very sensitive to the initial level set curve. Thus, we use the same initial level set curve for all cases. The initial curve can be seen in Figure 1.

First, let us look at the results for the image with no noise in Figure 2. For both cases, the convergence is much better than the geodesic active contour model in the last report. The non-convex Chan-Vese model terminated at 150 iteration and the convex Chan-Vese model terminated at 7 iteration only. However, both the non-convex and the convex Chan-Vese models seems to pick up some noise from the grass. We tried to tune the parameters but these noise from the grass did not appear to go away. After carefully examined the implementation details in Chan and Vese's original paper [1]. We found that the finite difference strategy is semi-implicit while we use a fully explicit in this report. We believe this can cause the undesired artifacts from the background. Now if we compare the results in Figures 2a and 2b, we can see clearly that the convex Chan-Vese picks up less artifacts from the grass, which can be explained as the susceptibility to local minima of the non-convex Chan-Vese. Moreover, the convex Chan-Vese converges much faster than the non-convex version.

Then we add noise to the image to examine the robustness of the algorithm. The results are shown in Figure 3. As expected, more artifacts are picked up from the background for both the non-convex



(a) (b)

Figure 1: Case with no noise. Initial level set curve for (a) the no noise image and (b) the noisy image.

and convex Chan-Vese model. Consistently, the convex Chan-Vese model performs better than the non-convex version in terms of the artifacts and convergence. From Figure 3d, we can see that the relative error of ϕ does not change much after around 30 iterations and simply oscillates, meaning we can even get almost the same result as appeared in Figure 3b after about 30 iterations.

4 Observation and Conclusions

The advantage of the Chan-Vese model over the snake model and geodesic active contour model is obvious.

- Edge detector is no longer needed, since Chan-Vese model deals with a different functional and the edge detector is not present.
- Chan-Vese model is not very sensitive to initial level set curve, though better initial curve should be able to speed up the convergence.
- Reinitialization is optional to Chan-Vese model. If applied, it should not be too strong.
- Overall, the convergence is also better than the edge-detection-based active contour models.

Furthermore, by comparing the non-convex and convex Chan-Vese models. We notice that the convex Chan-Vese model performs better in terms of both the artifacts in the background and the convergence.

In the end, one potential reason why our implementation of the Chan-Vese models pick up artifacts form the grass could be that we are using a semi-implicit strategy to solve for the gradient flow. A simple and effective remedy for this could be to apply some smooth kernel to the image before using Chan-Vese model.

References

[1] Tony F Chan and Luminita A Vese. Active contours without edges. *Image processing, IEEE transactions on*, 10(2):266–277, 2001.

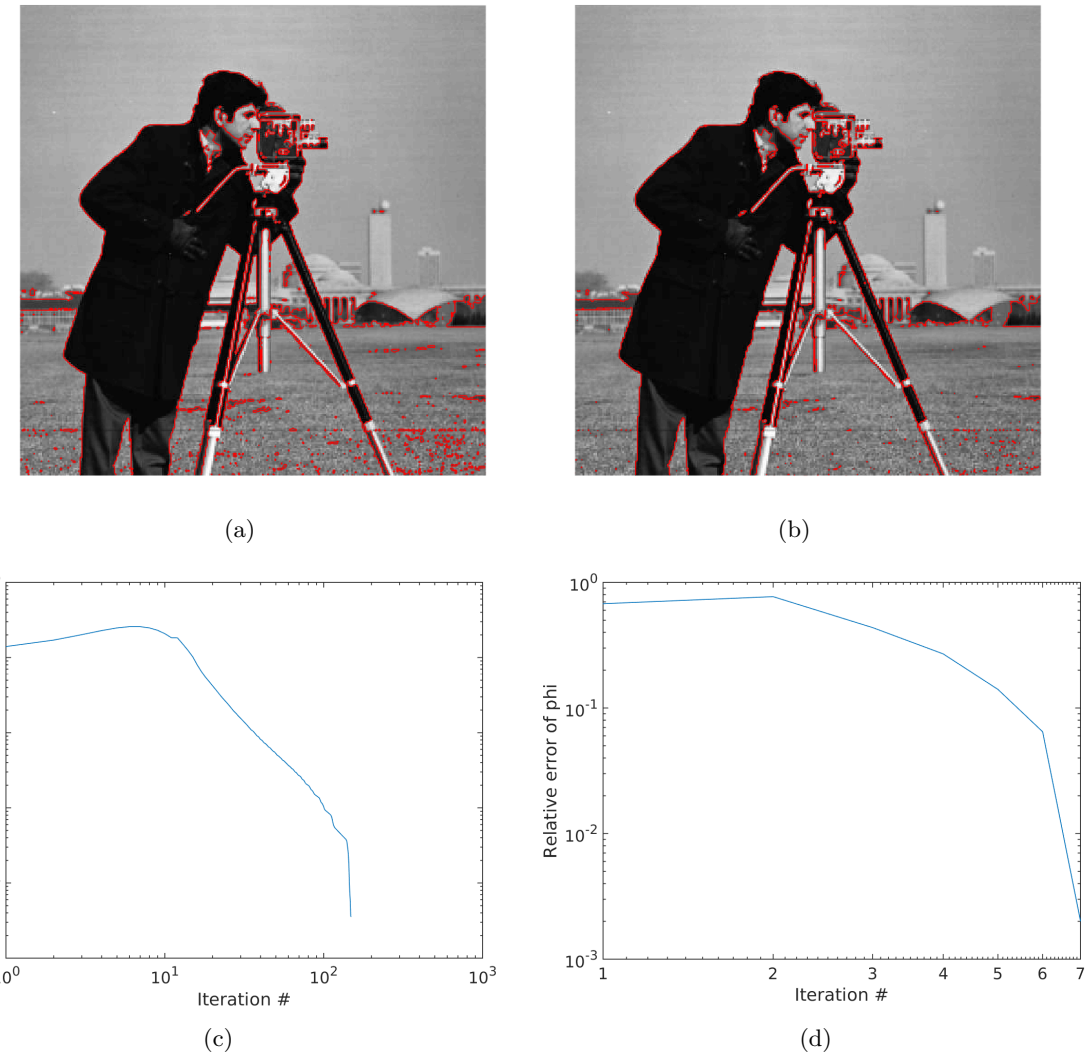


Figure 2: Case with no noise. Result for (a) the non-convex Chan-Vese model (b) the convex Chan-Vese model. Convergence plot for (c) the non-convex Chan-Vese model and (d) the convex Chan-Vese model where the vertical axis displays $\frac{||\phi^{n+1}-\phi^n||}{||\phi^n||}$.

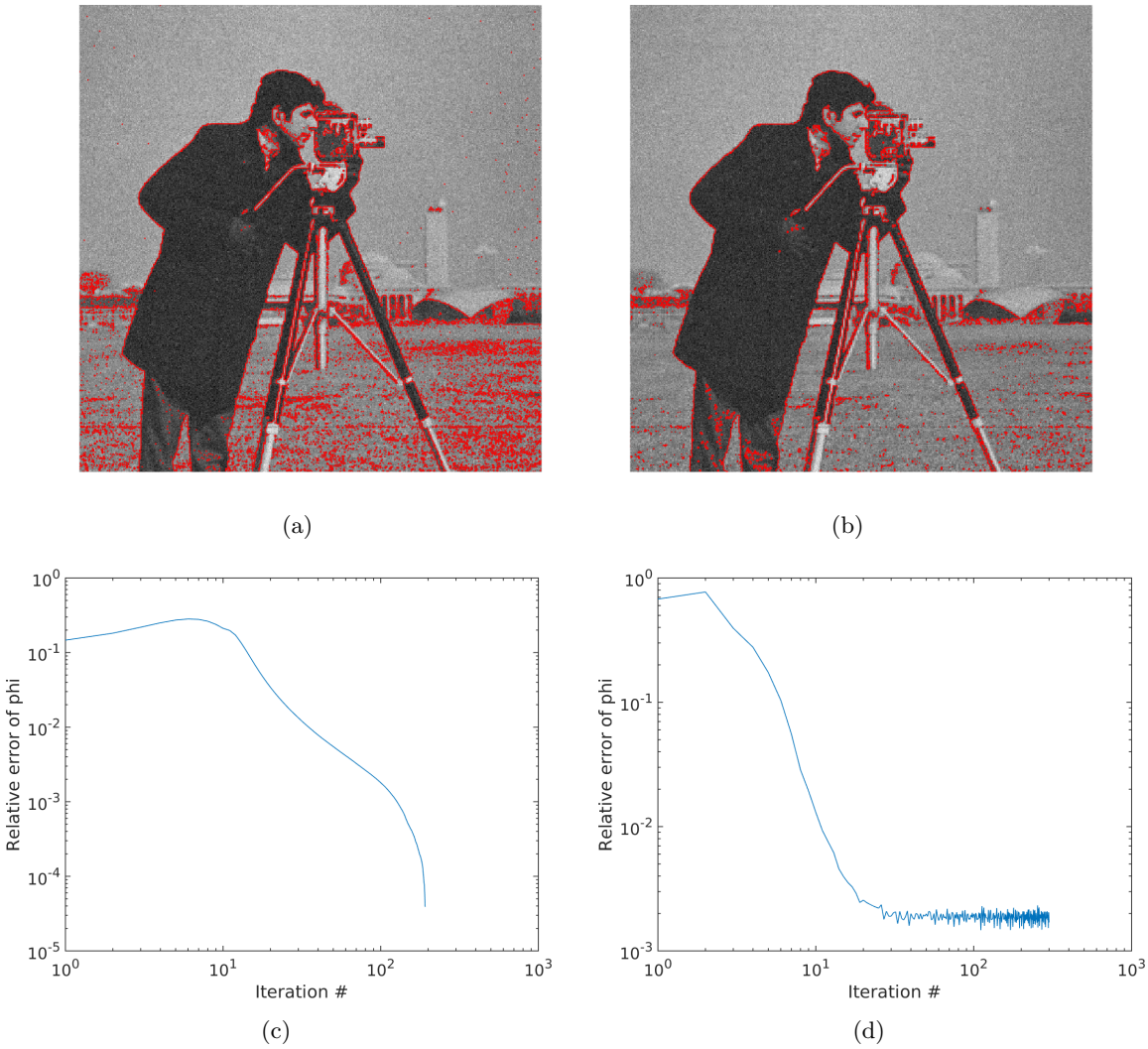


Figure 3: Case with noise. Result for (a) the non-convex Chan-Vese model (b) the convex Chan-Vese model. Convergence plot for (c) the non-convex Chan-Vese model and (d) the convex Chan-Vese model where the vertical axis displays $\frac{||\phi^{n+1}-\phi^n||}{||\phi^n||}$.

[2] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, 42(5):577–685, 1989.

[3] Tony F Chan, Selim Esedoglu, and Mila Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM journal on applied mathematics*, 66(5):1632–1648, 2006.

Appendix

MATLAB codes.

main.m specifies the parameters for the non-convex and convex Chan-Vese model, selects which method to use, calls the corresponding method subroutine and saves the results.

```
clear
close all

[I_0, I_n]=LoadData;

I=I_0;
iter=300;
beta=1e-8; % for smooth TV
dt=0.1;

method=2; % 1: non-convex CV; 2: convex CV

switch method
    case 1
        % non-convex Chan-Vese
        lambda1=0.1;
        lambda2=0.1;
        mu=1;
        nu=0;
        % smooth heaviside fun. and its derivative
        epsi=10; % for smooth delta function

        % heav = @(x) 0.5*(1+x/epsi+sin(pi*x/epsi)/pi).*(abs(x)<=epsi) + ...
        %             1.*(x>epsi) + 0.*(x<-epsi);
        % delta = @(x) 0.5*(1/epsi + cos(pi*y/epsi)/epsi).*(abs(x)<=epsi) + ...
        %              0.*(x>epsi) + 0.*(x<-epsi);

        heav = @(x) 0.5*(1+2*atan(x/epsi)/pi).*(abs(x)<=epsi) + ...
                    1.*(x>epsi) + 0.*(x<-epsi);

        delta = @(x) 1./(epsi*pi*((x/epsi).^2 + 1)).*(abs(x)<=epsi) + ...
                     0.*(x>epsi) + 0.*(x<-epsi);
    case 2
        % convex Chan-Vese
        lambda = 0.1;
        I1=0;
        I2=0;
        eta=0.2;
end
```

```
% initial level set
% rectangular
phi=zeros(size(I));
set0_row_top=71; set0_col_lef=47;
set0_row_bot=500; set0_col_rig=375;
phi(set0_row_top:set0_row_bot,set0_col_lef:set0_col_rig)=2;
phi(set0_row_top,set0_col_lef:set0_col_rig)=1;
phi(set0_row_bot,set0_col_lef:set0_col_rig)=1;
phi(set0_row_top:set0_row_bot,set0_col_lef)=1;
phi(set0_row_top:set0_row_bot,set0_col_rig)=1;
phi=phi-1;

h=figure;
conv_phi_fig=figure;
conv_obj_fig=figure;
conv_phi=zeros(iter,1);
conv_obj=zeros(iter,1);
mov_interval=10; % save 1 snapshot from each 10 iterations
mov(1:1 + iter/mov_interval)=struct('cdata',[],'colormap',[]);
mov_cnt=1;
for i=1:iter
    i

    figure(h);
    imagesc(I); colormap('gray'); axis off
    pbaspect([size(I),1]);
    hold on
    contour(phi,[1,1],'linecolor','red');
    drawnow

    if (i==1)
    saveas(h,sprintf('contour_0_%d.png',i));
    end

    phi_prev=phi;
    switch method
        case 1
            [phi,obj]=Chan_Vese(phi_prev,I,dt,heav,delta,mu,nu,lambda1,lambda2,beta);
        case 2
            [phi,I1,I2,obj]=convex_Chan_Vese_GD(phi_prev,I,I1,I2,lambda,beta,dt,eta);
    end


%     if (mod(i,10)==0)
%        phi=Reinitial2D(phi,10);
%        figure(h);
%        imagesc(I); colormap('gray'); axis off
%        pbaspect([size(I),1]);
%         hold on
%         contour(phi,[0,0],'linecolor','red');
%         drawnow
%     end

    if(mod(i,mov_interval)==0 || i==1)
```

```
        mov(mov_cnt)=getframe(h);
        mov_cnt=mov_cnt+1;
    end

    conv_obj(i)=obj;
    conv_obj(i)
    figure(conv_obj_fig);
    loglog(conv_obj); xlabel('Iteration #'); ylabel('Objective function value');
    set(gca,'fontsize',15)

    conv_phi(i)=norm(phi_prev(:)-phi(:))/norm(phi_prev(:));
    conv_phi(i)
    if(isnan(conv_phi(i)))
        display('Diverged!');
        break
    end
    figure(conv_phi_fig);
    loglog(conv_phi); xlabel('Iteration #'); ylabel('Relative error of phi');
    set(gca,'fontsize',15)

    if (i==iter)
        display('Done!');
        saveas(h,sprintf('contour_0_%d.png',i));
        saveas(conv_phi_fig,'conv_0_phi.png');
        saveas(conv_obj_fig,'conv_0_obj.png');
%         movie2avi(mov,'Chan_Vese_0.avi','compression','None');
    elseif (conv_phi(i)<=1e-10)
        display('Converged!');
        saveas(h,sprintf('contour_0_%d.png',i));
        saveas(conv_phi_fig,'conv_0_phi.png');
        saveas(conv_obj_fig,'conv_0_obj.png');
%         movie2avi(mov,'Chan_Vese_0.avi','compression','None');
        break
    end
end
%         figure; imagesc(heav(phi)); colormap(gray);
%         axis off
%         pbaspect([size(I),1]);
%         saveas(gcf,'bin_0.png');
```

Chan_Vese.m implements the non-convex Chan-Vese model.

```
function [phi,obj]=Chan_Vese(phi,I,dt,heav,delta,mu,nu,lambda1,lambda2,beta)

I1=trapz(1:size(phi,2),trapz(1:size(phi,1),I.*heav(phi),1)) / ...
   trapz(1:size(phi,2),trapz(1:size(phi,1),heav(phi),1));

I2=trapz(1:size(phi,2),trapz(1:size(phi,1),I.*(1-heav(phi)),1)) / ...
   trapz(1:size(phi,2),trapz(1:size(phi,1),1-heav(phi),1));

grad_phi_cc=grad(phi,[0 0]); % central
grad_phi_bb=grad(phi,[2 2]); % backward x, backward y
grad_phi_ff=grad(phi,[1 1]); % forward x, forward y
deno_x=sqrt(grad_phi_ff(:,:,1).^2+grad_phi_cc(:,:,2).^2+beta^2);
```

```
deno_y=sqrt(grad_phi_cc(:,:,1).^2+grad_phi_ff(:,:,2).^2+beta^2);
K=div(grad_phi_ff(:,:,1)./deno_x,grad_phi_ff(:,:,2)./deno_y);

phi = phi + dt*delta(phi).*( mu*K - nu -lambda1*(I-I1).^2 + lambda2*(I-I2).^2 );

grad_phi_cc=grad(phi,[0 0]); % central
grad_phi_tmp=sqrt(grad_phi_cc(:,:,1).^2+grad_phi_cc(:,:,2).^2).*delta(phi);
I1_tmp=(I-I1).^2.*heav(phi);
I2_tmp=(I-I2).^2.*(1-heav(phi));
heav_tmp=heav(phi);
obj=lambda1*sum(I1_tmp(:)) + lambda2*sum(I2_tmp(:)) + mu*sum(grad_phi_tmp(:)) + nu*sum(heav_tmp(:));
end
```

convex_Chan_Vese_GD.m implements the convex Chan-Vese model.

```
function [phi,I1,I2,obj]=convex_Chan_Vese_GD(phi,I,I1,I2,lambda,beta,dt,eta)

grad_phi_cc=grad(phi,[0 0]); % central
grad_phi_bb=grad(phi,[2 2]); % backward x, backward y
grad_phi_ff=grad(phi,[1 1]); % forward x, forward y
deno_x=sqrt(grad_phi_ff(:,:,1).^2+grad_phi_cc(:,:,2).^2+beta^2);
deno_y=sqrt(grad_phi_cc(:,:,1).^2+grad_phi_ff(:,:,2).^2+beta^2);
K=div(grad_phi_ff(:,:,1)./deno_x,grad_phi_ff(:,:,2)./deno_y);

force = K -lambda*(I-I1).^2 + lambda*(I-I2).^2;

phi = phi + dt*( force );
% thresholding
phi(phi>=1)=1; phi(phi<=0)=0;

Sigma = ones(size(I));
Sigma(phi<eta)=0;

I1=trapz(1:size(phi,2),trapz(1:size(phi,1),I.*Sigma,1)) / ...
   trapz(1:size(phi,2),trapz(1:size(phi,1),Sigma,1));

I2=trapz(1:size(phi,2),trapz(1:size(phi,1),I.*(1-Sigma),1)) / ...
   trapz(1:size(phi,2),trapz(1:size(phi,1),1-Sigma,1));

grad_phi_cc=grad(phi,[0 0]); % central
grad_phi_tmp=sqrt(grad_phi_cc(:,:,1).^2+grad_phi_cc(:,:,2).^2);
I_tmp = ( (I1-I).^2 - (I2-I).^2 ).*phi;
obj = sum(grad_phi_tmp(:)) + lambda*sum(I_tmp(:));

end
```

div.m implements the divergence operator.

```
function fd = div(Px,Py)
% backward x, backward y
nbdims = 2;
if size(Px,1)==1 || size(Px,2)==1
    nbdims = 1;
end
```

```
fx = Px-Px([1 1:end-1],:);
if nbdims>=2
    fy = Py-Py(:,[1 1:end-1]);
end

if nbdims==2
    fd = fx+fy;
else
    fd = fx;
end
```

grad.m implements the gradient operator.

```
function [fx] = grad(M,opt)
% 1:forward, 2:backward, 0:central

switch opt(1) % x dir
    case 1
        fx = M([2:end end],:)-M;
    case 2
        fx = M-M([1 1:end-1],:);
    case 0
        fx = ( M([2:end end],:) - M([1 1:end-1],:) )./2;
end

switch opt(2) % y dir
    case 1
        fy = M(:,[1 1:end-1])-M;
    case 2
        fy = M-M(:,[2:end end]);
    case 0
        fy = ( M(:,[2:end end])-M(:,[1 1:end-1]) )./2;
end

fx = cat(3,fx,fy);

end
```