Level Set Methods Implementation for Image Levelsets and Image Contour

NASSIR H. SALMAN,

Zarqa Private Univ., College of Science and Information Technology-Computer Science Dept. Jordan

Summary
Smoothing an image under the curvature of its level sets (Mean curvature flow), create contour plot of image data were used. To do that, a high level function (evolve2D) was used that can utilize any combination of: 1) a force in the normal direction to the curve, 2) a curvature-based force, and 3) evolution under the influence of an external vector field. This function takes an input, evolves it N iterations and returns the result. So; in this paper, all level sets of the image are smoothed under curvature-based forces and the level sets of the original image are calculated. Then the image was smoothed after 25-50 iterations of smoothing, and then we calculated its corresponding level sets. Also in this paper the Matlab function IMCONTOUR (img) was used to draw a contour plot of the intensity image (img). The method gives us accurately the level sets of our image before and after smoothing. The processing time for different processed images was calculated. Finally the results were compared with active contour method and levelset without re-initialization method.

Key words:
Levesets, Active contour, Curve evolution, Matlab, Image contour

1. Introduction

In two dimensions, the level set method amounts to representing a closed curve Γ in the plane as the zero level set of a two-dimensional auxiliary function ϕ,

$$\Gamma = \{(x, y) \mid \phi(x, y) = 0\}$$

and then manipulating Γ *implicitly*, through the function ϕ. This function is called a *level set function*. ϕ is assumed to take positive values inside the region delimited by the curve Γ and negative values outside. If the zero level set moves in the normal direction to itself with a speed *v*, this movement can be represented by means of a so-called *Hamilton-Jacobi equation* for the level set function:

$$\phi_t = v\left|\nabla \varphi\right|$$

This is a partial differential equation, and can be solved numerically

Manuscript received November 5, 2009
Manuscript revised November 20, 2009

In mathematics, a **level set** of a real-valued function *f* of *n* variables is a set of the form:[1]

$$\{ (x_1,...,x_n) \mid f(x_1,...,x_n) = c \} \quad ....(1)$$

where *c* is a constant. That is, it is the set where the function takes on a given constant value.When the number of variables is two, this is a level curve(contour line), if it is three (In 3-D or Surface (Interface) evolution), this is a level surface, and for higher values of *n* , the level set is a level hypersurface. Level Set Methods are used for the implementation of curve/interface evolution under various forces [2]. The advantage of the level set method is that: one can perform numerical computations involving curves and surfaces on a fixed cartesian grid without having to parameterize these objects (this is called the *Eulerian approach*). Also, the level set method makes it very easy to follow shapes that change topology, for example when a shape splits in two, develops holes, or the reverse of these operations see refs. [4], [5][6]

This article was distributed as follows: the introduction in section1 , the previous work in section 2, level set formulation of the model in section 3, the algorithm and the code are explanation in section4, the results are illustrated in section5 and finally the conclusion and discussion in section 6.

2. The Previous Work

In [2], the set of Matlab files implements Level Set Methods and follows Osher and Fedkiw's book was used. A combination of curvature-based forces, vector field-based forces and forces in the normal direction can be used. In [4] and [5] curve evolution, using Mumford–Shah functional for segmentation and level sets method to detect object(region) boundaries. In [6] Chunming Li , implemented level set evolution without re-initialization: A New Variational Formulation.

3. Level Set Formulation of The Model

In our work, the main idea for level set methods is: Evolve the embedding function *f(x,y)*, where image itself is taken as *f(x,y)* and keep track of its zero level set .This can thought as the diffusion of a 2-D surface.[2] see the next sections.

4. The Algorithm Explanation and Code

The algorithm was implemented using MATLAB 7.6.0(R2008a) [3], we modified and test the following functions using callback function, where ***Callback routine executed during object creation*** and **MATLAB executes the specified callback when a user activates a user interface control. The functions as follows:**

1-The high level function evolve2D() : this function was built separately and then callback . The form of this function is:[2]
function [phi] = evolve2D(phi, dx, dy, alpha , iterations, accuracy, is_signed_distance, normal_evolve, Vn, vector_evolve, u, v, kappa_evolve, b)

This function that takes an input as shown , evolves it N **iterations** and returns the result. Function (evolve2D) is provided that can utilize any combination of: 1) a force in the normal direction to the curve, 2) a curvature-based force, 3) evolution under the influence of an external vector field. In curvature diffusion (smoothing) image itself is taken as 2D *f(x,y)* and each level set of the image is evolved. The prototype and documentation for this function(input parameters) as follows:

-phi is the input level set function. **dx** and **dy** are the resolution of the grid at x and y dimensions.
-alpha is a constant for calculating the Euler step (*dt*). It should be between 0 and 1. 0.5 is quite safe whereas 0.9 can be risky.
-iterations specify the number of iterations before the function returns.
for other documentation , see Ref[2]
Where, all the functions such as (**is_signed_distance, normal_evolve, vector_evolve, u, v, kappa_evolve) are callback when we call function evolve2D(). There are built separately.** On other hand, before we used function phi **evolve2D()** above, we used **imcontour(phi)** and also we used it after using **evolve2D()** to get level sets of the origin image and the level sets after number of iterations ,see function **evolve2D()** definition above. In this paper, function **evolve2D ()** parameters were set as follows:

phi = evolve2D(phi,dx,dy,0.5,50,[],[],0,[],0,[],[],1,b);

2-The function reinit_SD() can be used for re-initialization of a signed distance function. This function would be especially effective if the input is not too far away from a signed distance function**.** The form of this function is:[2]
function [phi] = **reinit_SD**(phi, dx, dy, alpha, accuracy, iterations). And its parameters were set as follows:

%phi = reinit_SD(phi, dx, dy, 0.1, 'ENO3', 2);

Reinitializes **phi** into a signed distance function while preserving the zero level set (the interface or the curve. It is clear from the above two functions that function evolve2D() does not call reinit_SD() within itself (and as a result may not preserve signed distance-ness of phi). On the other hand, if desired, in an outside loop you can call evolve2D() for 25-50 iterations followed by a call to reinit_SD(), then call evolve2D() again for 25-50 more iterations, etc.[2]

In this case, all level sets of the image are smoothed under curvature-based forces. See Matlab code in the next section .Finally each level set of the image is evolved and the processing time depend upon iteration numbers and image size. Note that Matlab's contour function selects the levels automatically.

3-IMCONTOUR function : it was used to create contour plot of our image data, where, IMCONTOUR(I) draws a contour plot of the intensity image I, automatically setting up the axes , so their orientation and aspect ratio match the image.[3]

4-OUR MATHLAB CODES OF CALLING ABOVE FUNCTIONS:

% To open any image file format
```
[filename,              pathname]            =
uigetfile('*.bmp|*.jpg|*.gif|*.tif', 'Pick your image');
        if isequal(filename,0) || isequal(pathname,0)
disp('User pressed cancel')
        else
disp(['User selected ', fullfile(pathname, filename)])
   img = imread(filename)     // to read the image file
   end
imshow(img);                    // to display the image
dx=1;
dy=1;
tic;                    // to start calculate processing time
phi = double(img(:,:,1));
figure;
```

```
subplot(2,2,1); imagesc(phi); axis image; colormap
gray;
title('Origin image');
subplot(2,2,2); imcontour(phi); axis image;
title('Levelsets of the origin image');

% b: weighting for curvature-based force. b needs
to be positive
b = 0.3*ones(size(phi));

phi = evolve2D(phi,dx,dy,0.5,25,[],[],0,[],0,[],[],1,b);
// calling back  evolve2D() function

subplot(2,2,3); imagesc(phi); axis image; colormap
gray;
phi = reinit_SD(phi, dx, dy, 0.1, 'ENO3', 2);  //
calling reinit_SD() function

title('Smooth image after 25 iterations');
subplot(2,2,4);imcontour(phi); axis image;

title('Levelsets result after 25 iterations');
figure;
imcontour(img,3);
title('Origin image contour');

figure;
imcontour(phi); axis image;

title('Levelsets result after 25 iterations');
phi = double(img(:,:,1));
figure;
imcontour(phi); axis image;
title('Levelsets of the origin image');
toc;// to calculate the processing time
```

5. The Results

Our test results are illustrated in figure1 , figure 2 , figures 3 ,figure 4 and figure 5. The origin images are jpeg format and we can use any format and size (see the matlab code). The levelsets for the origin image and corresponding level sets are calculated using evolve2D() and other function illustrated in section 4(the algorithm explanation). These functions are built in section 5 (our Matlab code), then called and tested. After 25-50 iterations of smoothing the original image , the corresponding level sets (25 -50 iterations) and the processing time are calculated.

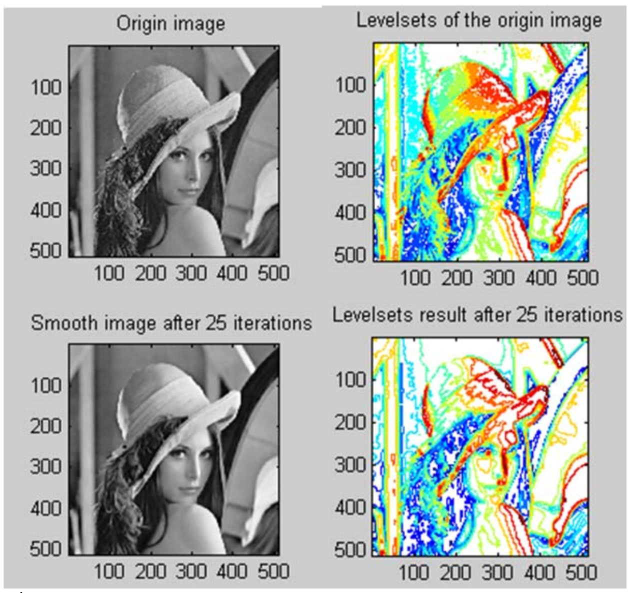
In figure 2 : Origin image 512x512 .jpg ,lena image. Elapsed time is 8.597722 seconds for evolution and elapsed time is **43.525709 seconds** for total processing. In figure 3 , the origin image 1024 x 768 .jpg , car color image, the elapsed time is **24.746785** seconds and elapsed time is **49.004153 seconds** for total processing. These results are compared with active contour technique results based on techniques of curve evolution, Mumford–Shah functional for segmentation and level sets method to detect object(region) boundaries[4], [5], see figure 4, also the results are compared with the levelset without re-initialization method [6] , see figure 5.

6. Conclosion and Discussion

By using and calling function evolve2D() we can smoothing an image under the curvature of its level sets (*Mean curvature flow*). The number of iterations depend upon the image type and size, and the evolution type depend upon the parameters values in function Phi
The functions we used in its current state can be used for understanding implementation of the Level Set Methods and implementing a 2-D curve evolution.
Number of iterations between 25-50 iterations are very good for our images, and evolution type depend upon the parameters values in function *Phi*. Also the processing time depend upon iteration numbers and image size. See section 6. The methods we used gave us accurately the level sets of our image before and after smoothing. The results of the levelset without re-initialization method [6] are also accurate. We can process any type of images such as (.**bmp, jpg, gif, tiff**,….).using the technique introduced in this paper. Finally, the results was compared with active contour technique results based on techniques of curve evolution, Mumford–Shah functional for segmentation and level sets method to detect object(region) boundaries[4], [5], see figure 4

References

[1] http://en.wikipedia.org/wiki/Level_set
[2] Baris Sumengen, A Matlab toolbox implementing Level Set Methods., vision research lab at UC Santa Barbara. 2005.
[3] Matlab the language of technical computing, version 7.6.0.324(R2008a)
[4] T. Can and L. Vese. Active contours with out edges. IEEE Trans. On image processing,10(2): 266-277 , 2001.
[5] N. H., SALMAN , C.Q., LIU . Active Contours and Mumford-Shah Segmentation Based on Level Sets. Shanghai Jiao Tong Univ. Journal. . Vol.E-8 , No1, June 2003.
[6] Chunming Li :"Level Set Evolution Without Re-initialization: A New Variational Formulation " in Proceedings of CVPR'05, vol. 1, pp. 430-436.2005


Origin image
Levelsets of the origin image
Smooth image after 25 iterations
Levelsets result after 25 iterations

1st example: Origin image 512x512 .jpg image format.
Figure 1 image levelset and image contour for origin image and after many iteration of evolving function. *Elapsed time is 8.597722 seconds for evolution.*
Elapsed time is 43.525709 seconds for total process of the above figures.

2nd example: Origin image 1024 x 768 .jpg

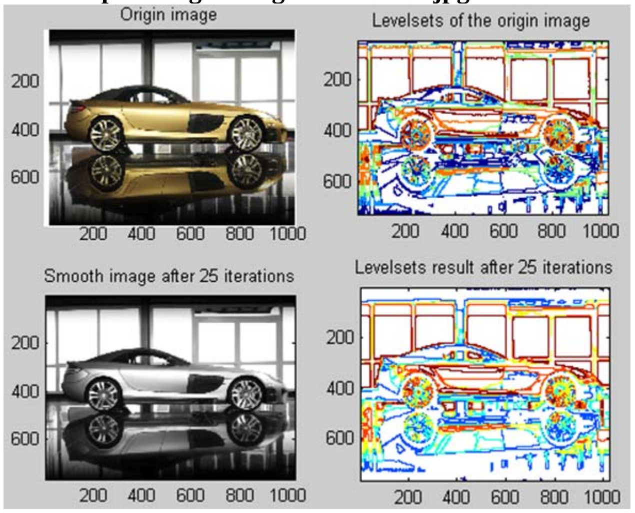

Origin image
Levelsets of the origin image
Smooth image after 25 iterations
Levelsets result after 25 iterations

Figure 2 image levelsets and image contour for origin image and after many iteration of evolving function.
3rd example brain image

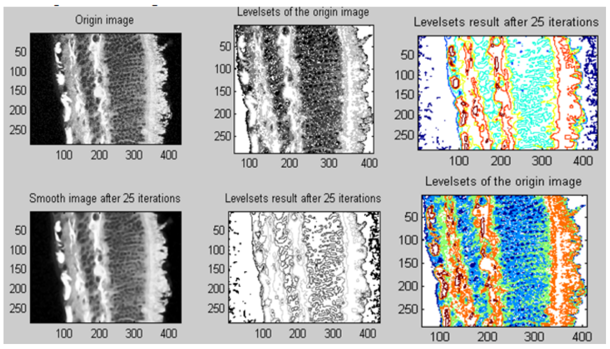

Origin image
Levelsets of the origin image
Levelsets result after 25 iterations
Smooth image after 25 iterations
Levelsets result after 25 iterations
Levelsets of the origin image

Figure 3 levelset for image brain and the elapsed time of processing(28.5 sec)

4th example: Active Contour results

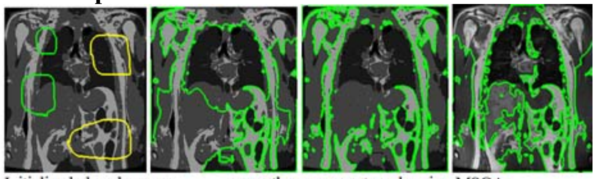
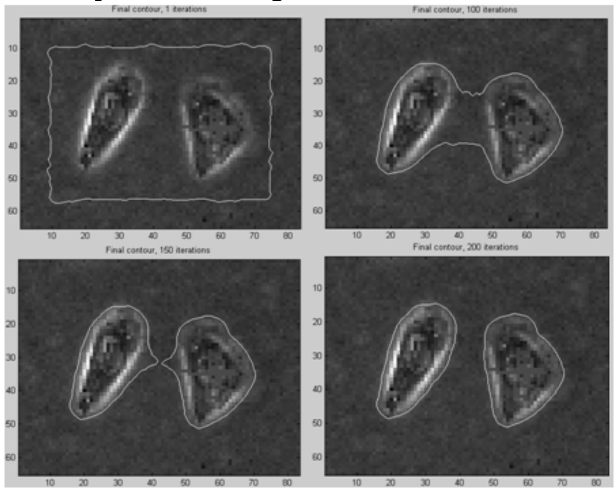

Initialized closed curves on origin image
grow the curves outwards using MSGA

Figure 4, shows initializing a small curve(s) inside the region(s) of interest, and allowing it to grow outwards until it reaches the desired boundary as follows: we set seed points inside every closed curve represent an intensity value of that region. After that we start initialized Mumford-shah geodesic active contours to propagate the initial seed point outwards, followed by a level set method to represent the curve in order to fine tune the result. See [5]. Other example of levelset methods as in [6].

5th example twocells image


Final contour, 1 iterations
Final contour, 100 iterations
Final contour, 150 iterations
Final contour, 200 iterations

Figure(5)"Level Set Evolution Without Re-initialization for twocells image ,see the number of iterations for each image above to segment the two cells in the image