ACTIVE CONTOUR METHOD TO SEGMENT TUMOR REGIONS IN MEDICAL IMAGES

Anup Puri, David. H. Frakes

School of Biological and Health Systems Engineering, Arizona State University, Tempe, AZ

ABSTRACT

Medical imaging has come a long way since the discovery of X-rays by Professor Roentgen in the first decade of 1900’s. The advent of the digital computer and new imaging modalities like ultrasound and magnetic resonance imaging (MRI) have combined to create an explosion of diagnostic imaging techniques in the last 3 decades. The key to using these techniques effectively is to be able to invent new modalities like ultrasound and magnetic resonance imaging (MRI) which results in various signal intensities; The signal intensities for different brain tissues; 4) Some scanning machines produce images with a low Signal to Noise Ratio (SNR) which leads to a poor image quality; 5) Field coil inhomogeneities cause intensity gradients across the scan image which perhaps would not affect analysis by a radiologist, but throws a threshold right out for automated segmentation algorithms; 6) Motion such as those caused in blood vessels and muscles causes noise or ringing around the movement; 7) Not all anatomical borders are intensity variations that can be easily segmented out. Anatomical borders are often soft boundaries that can’t be picked up by algorithms that rely on a thresholding approach.

The Chan-Vese model for active contours is a powerful and flexible method that is able to segment a variety of images, including those that would be especially difficult to segment, using the classical methods such as thresholding and utilizing gradients [1]. This model is based on the Mumford-Shah functional for segmentation, and is used widely in the medical imaging field, especially for the segmentation of the brain, heart & trachea [2] [3].

In this paper we implement the algorithm in MATLAB and test how it performs on MR images with different levels of the smoothness term.

Index Terms— Active contours, brain tumor segmentation, Chan-Vese algorithm, level sets.

1. INTRODUCTION

MRI scans are often required to take a close look at internal anatomical structures and arrive at a conclusive diagnosis for each patient. Numerous automated segmentation algorithms are being developed to assist radiologists in interpreting MR images. However, many a times, it becomes difficult to interpret medical images due to substandard imaging practices or due to poorly maintained imaging apparatus. Some difficulties encountered while using automated segmentation algorithms on MR images are: 1) Nearly similar intensities of tumor region and normal brain tissue; 2) Scans obtained from different MRI machines (or the same machine at different times) and for any given tissue type using different parameters produces different absolute intensities, different contrasts and also produces different overall scan quality; 3) Different brain structures have different tissue characteristics which results in various signal intensities; The signal intensities for different brain structures often overlap. Also, some non-brain tissues such as in the neck and scalp have the same intensities as brain tissues; 4) Some scanning machines produce images with a low Signal to Noise Ratio (SNR) which leads to a poor image quality; 5) Field coil inhomogeneities cause intensity gradients across the scan image which perhaps would not affect analysis by a radiologist, but throws a threshold right out for automated segmentation algorithms; 6) Motion such as those caused in blood vessels and muscles causes noise or ringing around the movement; 7) Not all anatomical borders are intensity variations that can be easily segmented out. Anatomical borders are often soft boundaries that can’t be picked up by algorithms that rely on a thresholding approach.

The objective of the Chan-Vese algorithm is to minimize the energy function $F(c_1, c_2, C)$, defined by:

$$F(c_1, c_2, C) = \mu \text{Length}(C) + \nu \text{Area insider}(C) + \lambda_1 \int |u_0(x,y) - c_1|^2 \, dx \, dy + \lambda_2 \int |u_0(x,y) - c_2|^2 \, dx \, dy$$

Equation 1: The energy functional

where, the first term stands for the energy inside C, and the second term for the energy outside C. Also, $\mu \geq 0, \nu \geq 0, \lambda_1, \lambda_2 \geq 0$ are fixed parameters (should be defined by the user). As suggested in the paper by Chan &
Vese, the preferred settings are \( v = 0 \), \( \lambda_1 = \lambda_2 = 1 \). We should note that the term \( \text{Length}(C) \) could be re-written more generally as \( (\text{Length}(C))^p \) for \( p \geq 1 \), but usually \( p = 1 \).

In other words we are looking for \( c_1, c_2, C \) that will be the solution to the minimization problem:

\[
\inf_{\delta_1, \delta_2, C} F(c_1, c_2, C)
\]

**Equation 2: The minimization problem**

### 2.1. Level Set formulation

Instead of searching for the solution in terms of \( C \), we can redefine the problem in the level set formalism. In the level set method, \( C \subseteq \Omega \) is represented by the zero level set of some Lipschitz function \( \phi : \Omega \to \mathbb{R} \), such that:

- \( C = \partial \omega = \{(x, y) \in \Omega : \Phi(x, y) = 0\} \)
- \( \text{inside}(C) = \omega = \{(x, y) \in \Omega : \Phi(x, y) > 0\} \)
- \( \text{outside}(C) = \Omega \setminus \overline{\omega} = \{(x, y) \in \Omega : \Phi(x, y) < 0\} \)

**Equation 3: Level set formulation**

![Image](image.png)

**Figure 1: The signed distance function**

Given a contour \( C, \Phi(x, y) \) is defined as the signed distance function from \( C \) where outside \( C \) the sign of \( \Phi(x, y) \) is negative. The objective is to evolve \( \Phi(x, y) \), when the evolved contour \( C \) in each time \( t \) is the zero level set of \( \Phi(x, y, t) \).

We can re-write the functional \( F(c_1, c_2, C) \) in terms of \( \Phi(x, y) \) only to obtain:

\[
\text{Length}(C) = \int_{\Omega} \| \nabla \Phi(x, y) \| \, dx \, dy = \int_{\Omega} \delta_1(\Phi(x, y)) \| \nabla \Phi(x, y) \| \, dx \, dy
\]

where \( H(z) \) is the Heaviside function \( H(z) = 1 \), for \( z \geq 0 \), and \( H(z) = 0 \), for \( z < 0 \).

\[
\text{Area}(\text{inside}(C)) = \int_{\Omega} H(\Phi(x, y)) \, dx \, dy
\]

\[
\Phi(x, y) \text{ is positive inside the closed curve } C.
\]

\[
\int_{\text{inside}(C)} |u_0(x, y) - c_1| \, dx \, dy = \int_{\text{inside}(C)} |u_0(x, y) - c_1| H(\Phi(x, y)) \, dx \, dy
\]

This is the energy function in terms of \( \phi(x, y) \) inside the curve \( C \), similarly for the energy function outside the curve \( C \), we get:

\[
\int_{\text{outside}(C)} |u_0(x, y) - c_2| \, dx \, dy = \int_{\text{outside}(C)} |u_0(x, y) - c_2| H(1 - \Phi(x, y)) \, dx \, dy
\]

We get the average intensities as:

\[
c_1 = \frac{\int_{\Omega} u_0(x, y) H(\Phi(x, y)) \, dx \, dy}{\int_{\Omega} H(\Phi(x, y)) \, dx \, dy}
\]

\[
c_2 = \frac{\int_{\Omega} u_0(x, y) H(1 - \Phi(x, y)) \, dx \, dy}{\int_{\Omega} H(1 - \Phi(x, y)) \, dx \, dy}
\]

The above leads us to the energy functional \( F \) in terms of \( (c_1, c_2, \phi) \), where \( c_1 = c_1(\phi) \), \( c_2 = c_2(\phi) \), and \( \delta_0(x) \) is the Dirac delta function.

\[
F(c_1, c_2, \phi) = \mu \int_{\Omega} \delta_1(\Phi(x, y)) \| \nabla \Phi(x, y) \| \, dx \, dy + \nu \int_{\Omega} H(\Phi(x, y)) \, dx \, dy
\]

\[
- \lambda_1 \int_{\Omega} |u_0(x, y) - c_1| H(\Phi(x, y)) \, dx \, dy - \lambda_2 \int_{\Omega} |u_0(x, y) - c_2| H(1 - \Phi(x, y)) \, dx \, dy
\]

**Equation 4: Energy formulation in terms of \( \phi \).**

Observing the terms in equation 4, we can say that the evolution of the curve is influenced by two terms (\( v \) is usually set to 0, so we will ignore it): the curvature regularizes the curve and makes it smooth during evolution; the “region term” \( -\lambda_1 (u_0 - c_1)^2 + \lambda_2 (u_0 - c_2)^2 \) affects the motion of the curve [4].

The term \( \lambda_1 \int_{\Omega} |u_0(x, y) - c_1| H(\Phi(x, y)) \, dx \, dy \) is the penalty on the total length of the curve \( C \). For example if the
boundaries of the image are quite smooth, we will give $\mu$ a larger value, to prevent $C$ from being a complex curve.

3. CODE IMPLEMENTATION

To validate the robustness of the code, we tested the algorithm on a few sample images as shown in [1]. We can clearly see that the algorithm perform well on these images.

Figure 2: Segmentation of an image (Right) with approximately two gray levels (Left).

Since, Chan-Vese algorithm is not based on gradient methods, and deals with balancing the energy on both sides of the curve, it achieves good results even when the image is blurred (as seen in Figure 3). One of the main advantages of the algorithm is that it relies on global properties, rather than just taking into account local properties, such as gradients. This provides better robustness for noise, as is clearly seen from the next set of sample images.

Figure 3: Left: blurred objects, Right: segmentation.

Figure 4: Left: No noise, Middle: modest noise, Right: stronger noise.

4. TEST DATA

Test data was obtained from Barrow Neurological Institute, Phoenix, and we tested the algorithm written in MATLAB on these test images. The test data acquired were Magnetic Resonance image sequences acquired from 11 patients whose identities were unknown. We tested the Chan-Vese algorithm on these test images by varying the smoothness term in the MATLAB code to achieve better segmentation when the image quality was inferior (Attached as Appendix – I).

4.1. Working with the test data.

The test image to be processed for the tumor segmentation is loaded into MATLAB, and we use the region of interest selection tool in MATLAB (using `imellipse`), and are able to create masks from the test image. This mask serves as the initial curve for the algorithm, and algorithm then begins to evolve in the neighborhood to obtain the final segmentation based on the properties of the objects in the image.

Figure 5: Left: 12th slice in a MRI sequence, Right: selecting probable tumor area as region of interest.

5. RESULTS

The masked test data is introduced to the region segmentation algorithm and the results for the three values of the smoothness term is presented below. The three values of the smoothness term chosen for the above slice are $0.7$, $0.3$ & $0.1$. The algorithm also gives out a binary mask after the segmentation process is over. We ran the algorithm on the test data for 700 iterations, as the image is a large 256 X 256 image. A larger value of the smoothness term imposes a rigidity on the evolving curve. The curve covers more variability and is more dynamic for smaller values of the smoothness term.
We can clearly see the best result is available for the smoothness term at a value of 0.1. This can be better seen when we take a look at the binary images that are output from the algorithm.

Binary masks from the algorithm give us a better idea of the shape of the tumor region in the above slice. We can see that the best segmentation is achieved in the right most image in Figure 7, when comparing the shape of the tumor in the original image from Figure 5.

The results for various slices were achieved in the same fashion, and the algorithm performed really well on the test data set overall.

**6. FUTURE WORK**

We need to work on pre-processing algorithms in order to make the input images into the algorithm uniform, in terms of the level of noise they contain and the intensity ratios that each image has in the MR slice sequence. The value of the smoothness term can be kept constant or it can be adapted to fixed noise-level and intensity parameters for the entire MR sequence. This makes the algorithm more powerful with respect to the correctness of the output, and increases overall reliability.

This work can be extended further to segment tumor regions in each slice of a MRI sequence. With such valuable information it is possible to create a 3-D rendition of the tumor using image registration software such as 3D- Doctor or Osirix. Such 3-D shapes when superimposed on the volumetric rendition of the entire MR sequence, can give the doctors a clear idea of the location of a tumor in 3-D space.

Such information comes in handy during the time of surgical planning specially for sensitive areas as the Human Brain or the Spinal Cord.

**7 CONCLUSION**

Chan-Vese algorithm was implemented in this project. From the results above, it can be seen that this algorithm deals quite well even with images which are quite difficult to segment in the regular methods, such as gradient-based methods or thresholding. This can be explained by the fact that Chan-Vese algorithm relies on global properties (intensities, region areas), rather than just taking into account local properties, such as gradients. One of the main advantages of this approach is better robustness for noise. This approach of the algorithm is especially useful for the medical community if the algorithm is efficient and reliable. Sometimes, it becomes really difficult for the doctors to segment out regions of interest from MR or CT data due to the complexity of the anatomical feature involved or due to nearly similar intensities of neighboring tissue. Such problems can be solved by utilizing the algorithm for the dataset and reliable results can be achieved.

**8. REFERENCES**


% Region Based Active Contour Segmentation
%
% seg = region_seg(I, init_mask, max_its, alpha, display)
%
% Inputs: I           2D image
%         init_mask   Initialization (1 = foreground, 0 = bg)
%         max_its     Number of iterations to run segmentation for
%         alpha       (optional) Weight of smoothing term
%                       higer = smoother. default = 0.2
%         display     (optional) displays intermediate outputs
%                       default = true
%
% Outputs: seg        Final segmentation mask (1=fg, 0=bg)
%
%------------------------------------------------------------------------
function seg = region_seg(I, init_mask, max_its, alpha, display)

%-- default value for parameter alpha is .2
if(~exist('alpha', 'var'))
    alpha = .2;
end
%-- default behavior is to display intermediate outputs
if(~exist('display', 'var'))
    display = true;
end
%-- ensures image is 2D double matrix
I = im2graydouble(I);

%-- Create a signed distance map (SDF) from mask
phi = mask2phi(init_mask);

%--main loop
for its = 1:max_its  % Note: no automatic convergence test
    idx = find(phi <= 1.2 & phi >= -1.2);  %get the curve's narrow band

    %-- find interior and exterior mean
    upts = find(phi<0);  % interior points
    vpts = find(phi>0);  % exterior points
    u = sum(I(upts))/(length(upts)+eps);  % interior mean
    v = sum(I(vpts))/(length(vpts)+eps);  % exterior mean

    F = (I(idx)-u).^2-(I(idx)-v).^2;  % force from image information
    curv = get_curvature(phi, idx);  % force from curvature penalty

    dphidt = F./max(abs(F)) + alpha*curv;  %gradient descent to minimize energy

    %-- maintain the CFL condition
    dt = .45/(max(dphidt)+eps);
    %-- evolve the curve
    phi(idx) = phi(idx) + dt.*dphidt;

    %-- Keep SDF smooth
    phi = sussman(phi, .5);
if((display>0)&&(mod(its,20) == 0))
    showCurveAndPhi(I,phi,its);
end

%-- final output
if(display)
    showCurveAndPhi(I,phi,its);
end

%-- make mask from SDF
seg = phi<=0; %-- Get mask from levelset

%-- AUXILIARY FUNCTIONS ----------------------------------------------

%-- Displays the image with curve superimposed
function showCurveAndPhi(I, phi, i)
    imshow(I, 'initialmagnification',200,'displayrange',[0 255]); hold on;
    contour(phi, [0 0], 'g', 'LineWidth',4);
    contour(phi, [0 0], 'k', 'LineWidth',2);
    hold off; title([num2str(i) ' Iterations']); drawnow;

%-- converts a mask to a SDF
function phi = mask2phi(init_a)
    phi=bwdist(init_a)-bwdist(1-init_a)+im2double(init_a)-.5;

%-- compute curvature along SDF
function curvature = get_curvature(phi,idx)
    [dimy, dimx] = size(phi);
    [y x] = ind2sub([dimy,dimx],idx); % get subscripts

%-- get subscripts of neighbors
    yml = y-1; xml = x-1; ypl = y+1; xpl = x+1;

%-- bounds checking
    yml(yml<1) = 1; xml(xml<1) = 1;
    ypl(ypl>dimy) = dimy; xpl(xpl>dimx) = dimx;

%-- get indexes for 8 neighbors
    idup = sub2ind(size(phi),ypl,x);
    iddn = sub2ind(size(phi),yml,x);
    idlt = sub2ind(size(phi),y,xml);
    idrt = sub2ind(size(phi),y,xpl);
    idul = sub2ind(size(phi),ypl,xml);
    idur = sub2ind(size(phi),ypl,xpl);
    iddl = sub2ind(size(phi),xml,xml);
    iddr = sub2ind(size(phi),xml,xpl);

%-- get central derivatives of SDF at x,y
    phi_x = -phi(idlt)+phi(idrt);
    phi_y = -phi(iddn)+phi(idup);
    phi_xx = phi(idlt)-2*phi(idx)+phi(idrt);
    phi_yy = phi(iddn)-2*phi(idx)+phi(idup);
    phi_xy = -0.25*phi(iddl)-0.25*phi(idur)...
\[ +0.25 \phi(i(ddr)+0.25 \phi(idul) \]
\[ \phi_x^2 = \phi_x \cdot \phi_x; \]
\[ \phi_y^2 = \phi_y \cdot \phi_y; \]

`%-- compute curvature (Kappa)`
\[ \text{curvature} = \frac{\phi_x^2 \cdot \phi_y y + \phi_y^2 \cdot \phi_x x - 2 \cdot \phi_x \cdot \phi_y \cdot \phi_{xy}}{(\phi_x^2 + \phi_y^2 + \text{eps})^{3/2}} \cdot (\phi_x^2 + \phi_y^2)^{1/2}; \]

`%-- Converts image to one channel (grayscale) double function` im2graydouble(img)
\[
[\text{dimy}, \text{dimx}, \text{c}] = \text{size(img)};
\]
\[
\text{if} \quad (\text{isfloat(img)}) \% \text{image is a double}
\]
\[
\quad \text{if} \quad (\text{c}==3)
\quad \quad \text{img} = \text{rgb2gray} (\text{uint8(img))};
\quad \text{end}
\]
\[
\text{else} \% \text{image is a int}
\quad \text{if} \quad (\text{c}==3)
\quad \quad \text{img} = \text{rgb2gray} (\text{img});
\quad \text{end}
\quad \text{img} = \text{double(img)};
\quad \text{end}
\]

`%-- level set re-initialization by the sussman method function` D = sussman(D, dt)
\[
\% \text{forward/backward differences}
\quad a = \text{D} - \text{shiftR(D)}; \% \text{backward}
\quad b = \text{shiftL(D)} - \text{D}; \% \text{forward}
\quad c = \text{D} - \text{shiftD(D)}; \% \text{backward}
\quad d = \text{shiftU(D)} - \text{D}; \% \text{forward}
\]
\[
\quad a_p = a; \quad a_n = a; \% a+ and a-
\quad b_p = b; \quad b_n = b;
\quad c_p = c; \quad c_n = c;
\quad d_p = d; \quad d_n = d;
\]
\[
\quad a_p(a < 0) = 0;
\quad a_n(a > 0) = 0;
\quad b_p(b < 0) = 0;
\quad b_n(b > 0) = 0;
\quad c_p(c < 0) = 0;
\quad c_n(c > 0) = 0;
\quad d_p(d < 0) = 0;
\quad d_n(d > 0) = 0;
\]
\[
\text{dD} = \text{zeros(size(D))};
\quad \text{D_neg_ind} = \text{find(D < 0)};
\quad \text{D_pos_ind} = \text{find(D > 0)};
\quad \text{dD(D_pos_ind)} = \sqrt{\text{max(a_p(D_pos_ind).^2, b_n(D_pos_ind).^2)} + \text{max(c_p(D_pos_ind).^2, d_n(D_pos_ind).^2}) - 1};
\quad \text{dD(D_neg_ind)} = \sqrt{\text{max(a_n(D_neg_ind).^2, b_p(D_neg_ind).^2)} + \text{max(c_n(D_neg_ind).^2, d_p(D_neg_ind).^2}) - 1};
\]
\[
\text{D} = \text{D} - \text{dt} \cdot \text{sussman_sign(D) .} \cdot \text{dD};
\]
%% whole matrix derivatives
function shift = shiftD(M)
    shift = shiftR(M')';

function shift = shiftL(M)
    shift = [ M(:,2:size(M,2)) M(:,size(M,2)) ];

function shift = shiftR(M)
    shift = [ M(:,1) M(:,1:size(M,2)-1) ];

function shift = shiftU(M)
    shift = shiftL(M')';

function S = sussman_sign(D)
    S = D ./ sqrt(D.^2 + 1);