به نام خدا

مینی‌پروژه ۴

مرز های فعال بدون لبه

Active Contour Without Edges

Chan – Vese

سید سجاد ائمی

شماره دانشجویی: ۹۶۱۳۶۶۶۱۱۹

استاد: دکتر مهدی سعادتمند طرزجان

تاریخ تحویل: ۹۷/۴/۲۰

توضیحات:

در الگوریتم Chan - Vese، علاوه بر تصویر اصلی که می خواهیم آن را تقطیع نماییم، یک تصویر دیگر به نام مدل داریم. مدل از نظر ابعاد دقیقا برابر با تصویر اصلی می باشد. و به دو ناحیه داخلی و خارجی تقسیم شده است که مقادیر ناحیه داخلی و خارجی آن، برابر با میانگین وزن دار پیکسل های داخل منحنی و میانگین وزن دار پیکسل های خارج منحنی می باشد. بر این اساس باید ابتدا تابع انرژی را بدست آوریم و سپس به کمک اویلر - لاگرانژ، معادله تکامل آن را محاسبه نماییم.

بدین ترتیب، مرز اولیه به سمت ناحیه داخل تصویر همگرا شده و عمل تقطیع انجام می پذیرد.

مرحله اول:

ابتدا تصویر رنگی را به تصویر خاکستری تبدیل می کنیم. سپس مقادیر سطح خاکستری پیکسل های آن را بین 0 و 1 تنظیم می کنیم.

```
if size(Img,3) == 3
    I = rgb2gray(Img);
    I = double(I);
else
    I = double(Img);
end
```

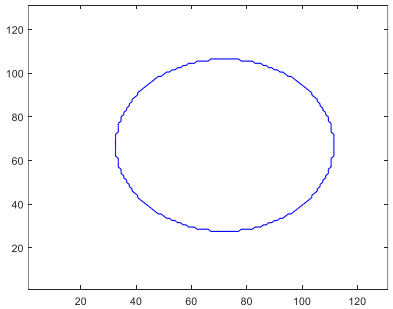
مرحله دوم:

در این قسمت یک مدل اولیه طراحی می کنیم. همانطور که در بالا توضیح داده شد، اندازه ابعاد مدل باید با تصویر اصلی برابر باشد. مدل اولیه می تواند دایره، مربع یا هر شکل دیگری باشد. در این پروژه از مدل اولیه دایروی استفاده شده است. در قطعه کد زیر یک مدل دایره ای با شعاع ۱۰ ایجاد شده است.

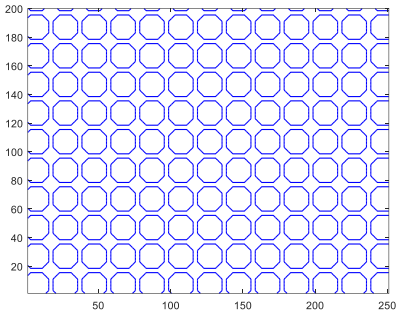
```
r = 10;
n = zeros(size(x));
n((x-cx).^2 + (y-cy).^2 < r.^2) = 1;

model = zeros(size(Img));
model(1 : size(n,1), 1 : size(n,2)) = n;
```

خروجی قطعه کد بالا به صورت زیر می باشد:



همچنین می توان مرز اولیه را به صورت تعدادی دایره کوچک تر در نظر گرفت:



مرحله سوم:

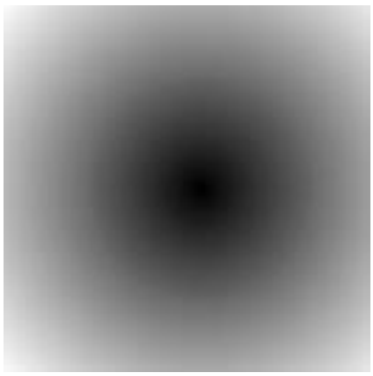
در این قسمت از روی مدلی که در مرحله قبل به دست آوردیم، φ_0 را طبق رابطه زیر محاسبه می کنیم:

$$\varphi_0 = Dt(model) - Dt(1 - model)$$

کد آن به صورت زیر می باشد:

```
phi = bwdist(Model) - bwdist(1 - Model);
```

خروجی این کد برای مدل اولیه دایره، به صورت زیر می باشد. که در خارج منحنی دارای مقادیر مثبت و داخل منحنی دارای مقادیر منفی می باشد.



مرحله چهارم:

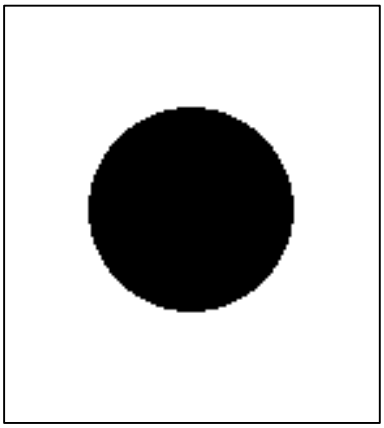
در این قسمت مقدار تابع $H_\varepsilon(\varphi)$ را طبق رابطه زیر محاسبه می کنیم. این تابع به دلیل نرم بودن در لبه ها، خاصیت مشتق پذیری دارد.

$$H_\varepsilon(\varphi) = \frac{1}{2} \times (1 + \frac{2}{\pi} \arctan(\frac{\varphi}{\varepsilon}))$$

کد آن به صورت زیر می باشد:

```
H_eps = 0.5 * (1 + (2/pi) * atan(phi./eps));
```

خروجی این کد به صورت زیر می باشد:



مرحله پنجم:

در این قسمت مقدار میانگین وزن دار داخل و خارج منحنی را محاسبه می کنیم.

$$C_1 = \frac{\int I(\underline{X}) H_\varepsilon(\varphi) d\underline{X}}{\int H_\varepsilon(\varphi) d\underline{X}} \qquad C_2 = \frac{\int I(\underline{X})(1-H_\varepsilon(\varphi)) d\underline{X}}{\int (1-H_\varepsilon(\varphi)) d\underline{X}}$$

کد آن به صورت زیر می باشد:

```
c1 = sum(sum(I .* H_eps)) / sum(sum(H_eps));
c2 = sum(sum(I .* (1 - H_eps))) / sum(sum((1 - H_eps)));
```

مرحله ششم:

در این قسمت نیروی داخلی و نیروی خارجی را محاسبه می کنیم و سپس نیروی کل را بدست می آوریم. رابطه نیروی داخلی و خارجی به صورت زیر می باشند:

$$F_{int} = \lambda_2 (I - C_2)^2 - \lambda_1 (I - C_1)^2$$

$$F_{ext} = \mu \times \kappa(\varphi) = \mu \times \nabla \cdot \frac{\nabla \varphi}{|\nabla \varphi|}$$

$$\frac{\partial C}{\partial \varphi} = F_{int} + F_{ext}$$

کد آن به صورت زیر می باشد:

```
F_int = lambda2 * (I-c2).^2 - lambda1 * (I-c1).^2;

F_ext = mu * kappa(phi);

F = F_int + F_ext;
```

مقدار λ_1 و λ_2 برابر ۱ در نظر گرفته شده است. و تابع $\kappa(\varphi)$ به صورت زیر می باشد:

```
function k = kappa(I)

    I = double(I);

    [Model_G_X ,Model_G_Y]  = gradient(I);
    Norm = sqrt(Model_G_X.^2 + Model_G_Y.^2 + eps);
    Model_NG_X = Model_G_X ./ Norm;
    Model_NG_Y = Model_G_Y ./ Norm;

    [Model_G_XX ,~] = gradient(Model_NG_X);
    [~ , Model_G_YY] = gradient(Model_NG_Y);

    Divergence  = Model_G_XX + Model_G_YY;
    k = Divergence;
end
```

مرحله هفتم:

در این قسمت معادله تکامل منحنی را نوشته و آن را آپدیت می کنیم.

$$\varphi^{i+1} = \varphi^{i} + \eta \frac{\partial C}{\partial \varphi}$$

کد آن به صورت زیر می باشد:

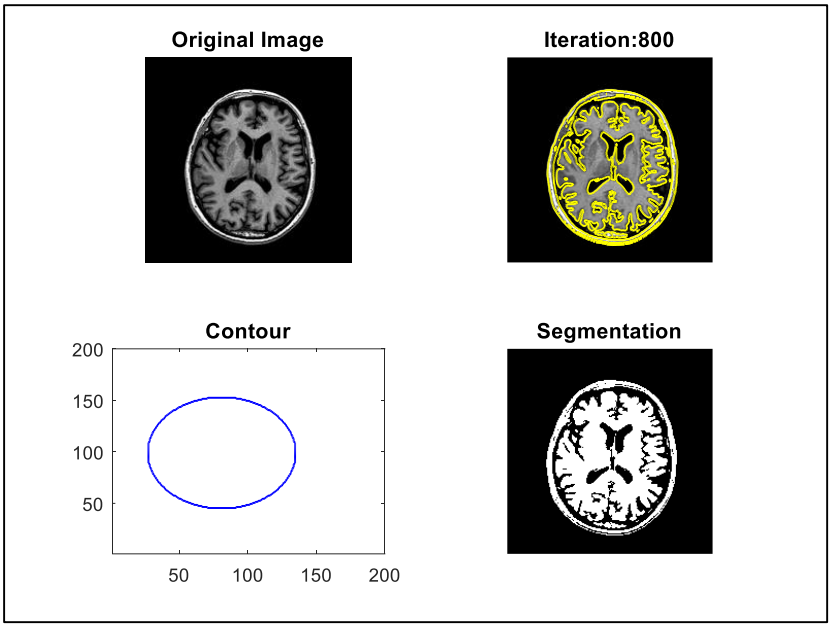
```
phi = phi + eta .* F;
```

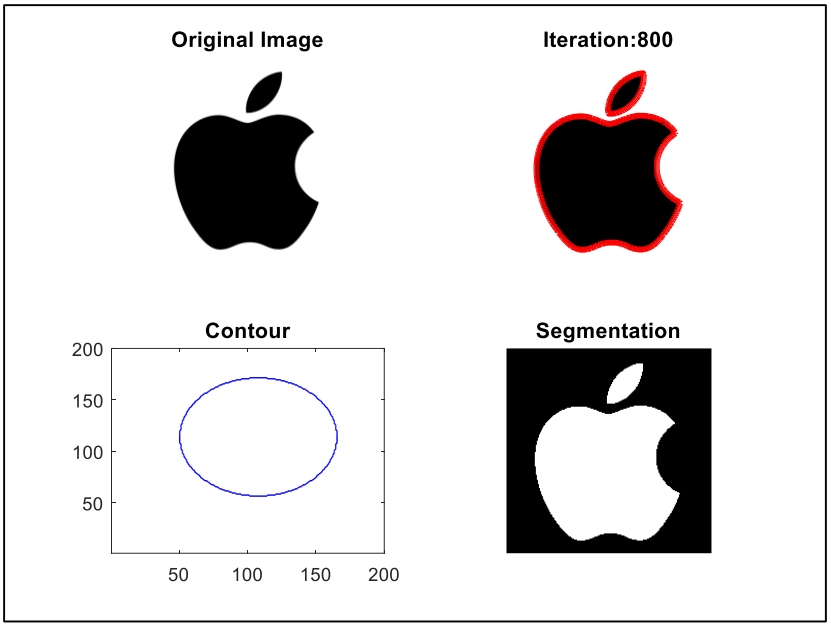
نتایج:

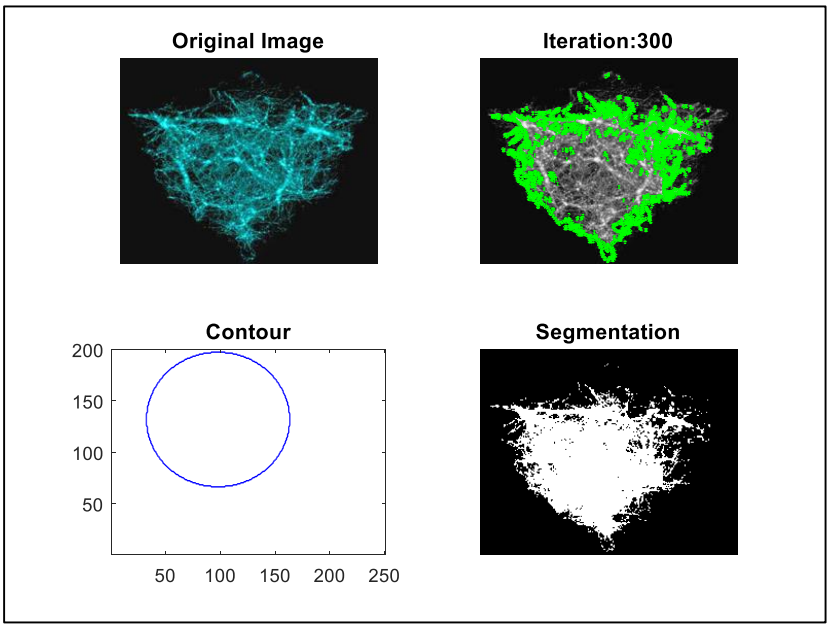
پارامتر ها را به صورت زیر تنظیم می کنیم:

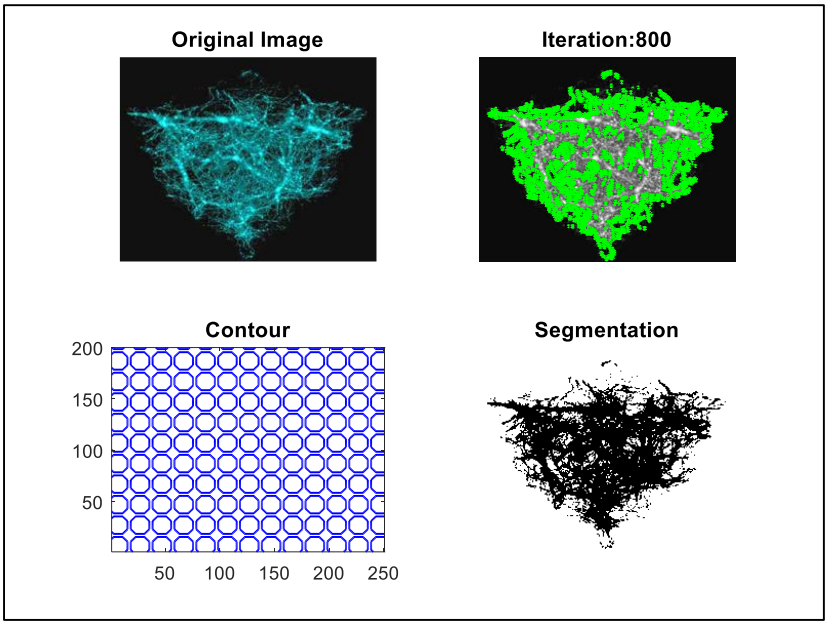
```
eta = 0.5;
lambda1 = 1;
lambda2 = 1;
mu = 0.02;
iteration = 800;
```

نتیجه الگوریتم Chan - Vese برای تصاویر آزمایشی به صورت زیر می باشد:


Original Image
Iteration:800
Contour
Segmentation
200
150
100
50
50
100
150
200


Original Image
Iteration:800
Contour
Segmentation
200
150
100
50
50
100
150
200


Original Image
Iteration:300
Contour
Segmentation
200
150
100
50
50
100
150
200
250



همانطور که مشاهده می شود، اگر مرز اولیه را به صورت تعداد زیادی دایره کوچک در نظر بگیریم، خروجی بهتری خواهیم داشت.

پیوست:

فایل اصلی main.m

```
% "Active Contours Without Edges" by Chan and Vese

close all
clear all

Img = imread('images/mri.jpg');

%resize original image
scale = 200./min(size(Img,1),size(Img,2));
if scale < 1
   Img = imresize(Img,scale);
end

seg = chanvese(Img,'medium',800,0.02);

Img = imread('images/apple.jpg');

%resize original image
scale = 200./min(size(Img,1),size(Img,2));
if scale < 1
   Img = imresize(Img,scale);
end

seg = chanvese(Img,'medium',800,0.02);
```

```
Img = imread('images/anti-mass.jpg');

%resize original image
scale = 200./min(size(Img,1),size(Img,2));
if scale < 1
   Img = imresize(Img,scale);
end

seg = chanvese(Img,'large',300,0.02);
seg = chanvese(Img,'whole',800,0.02);
```

تابع chanvese.m

```
function result = chanvese(Img, Model, iteration, mu)

%Grayscale Image
if size(Img,3) == 3
    I = rgb2gray(Img);
    I = double(I);
else
    I = double(Img);
end

%Create Model
switch lower (Model)
  case 'small'
      Model = circleModel(Img,'small');
  case 'medium'
      Model = circleModel(Img,'medium');
  case 'large'
      Model = circleModel(Img,'large');
  case 'whole'
      Model = circleModel(Img,'whole');
end

Model = Model(:,:,1);

%Hyper Parameters
eta = 0.5; %stepsize
lambda1 = 1;
lambda2 = 1;

phi = bwdist(Model) - bwdist(1 - Model);

figure;
subplot(2,2,1);
imshow(Img);
title('Original Image');
```

```
subplot(2,2,3);
contour(phi, [0 0], 'b','LineWidth',1);
title('Contour');

subplot(2,2,2);
imshow(I,[],'initialmagnification','fit');
hold on;

for i = 1:iteration

  H_eps = 0.5 * (1 + (2/pi) * atan(phi./eps));

  c1 = sum(sum(I .* H_eps)) / sum(sum(H_eps)); % average inside
  c2 = sum(sum(I .* (1 - H_eps))) / sum(sum((1 - H_eps))); % average outside

  F_int = lambda2 * (I-c2).^2 - lambda1 * (I-c1).^2;

  F_ext = mu * kappa(phi);
  F_ext = F_ext ./ max(max(abs(F_ext))); % normalize

  F = F_int + F_ext;
  F = F./(max(max(abs(F)))); % normalize

  phi = phi + eta .* F;

  cla;
  imshow(I,[],'initialmagnification','fit');
  contour(phi, [0 0], 'g','LineWidth',2);
  title(strcat('Iteration: ', num2str(i)));
  drawnow;

end

result = phi <= 0;

subplot(2,2,4);
imshow(result);
title('Segmentation');
end
```

تابع kappa.m

```
function k = kappa(I)
```

```
    I = double(I);

    [Model_G_X ,Model_G_Y]  = gradient(I);
    Norm = sqrt(Model_G_X.^2 + Model_G_Y.^2 + eps);
    Model_NG_X = Model_G_X ./ Norm;
    Model_NG_Y = Model_G_Y ./ Norm;

    [Model_G_XX ,~] = gradient(Model_NG_X);
    [~ , Model_G_YY] = gradient(Model_NG_Y);

    Divergence  = Model_G_XX + Model_G_YY;
    k = Divergence;
end
```

تابع circleModel.m

```
function model = circleModel(I,type)

if size(I,3)~=3
    temp = double(I(:,:,1));
else
    temp = double(rgb2gray(I));
end

h = [0 1 0; 1 -4 1; 0 1 0];
T = conv2(temp,h);
T(1,:) = 0;
T(end,:) = 0;
T(:,1) = 0;
T(:,end) = 0;

thre = max(max(abs(T)))*.5;
idx = find(abs(T) > thre);
[cx,cy] = ind2sub(size(T),idx);
cx = round(mean(cx));
cy = round(mean(cy));

[x,y] = meshgrid(1:min(size(temp,1),size(temp,2)));

model = zeros(size(temp));
[p,q] = size(temp);

switch lower (type)
    case 'small'
        r = 10;
```

```
        n = zeros(size(x));
        n((x-cx).^2+(y-cy).^2<r.^2) = 1;
        model(1:size(n,1),1:size(n,2)) = n;
    case 'medium'
        r = min(min(cx,p-cx),min(cy,q-cy));
        r = max(2/3*r,25);
        n = zeros(size(x));
        n((x-cx).^2+(y-cy).^2<r.^2) = 1;
        model(1:size(n,1),1:size(n,2)) = n;
    case 'large'
        r = min(min(cx,p-cx),min(cy,q-cy));
        r = max(2/3*r,60);
        n = zeros(size(x));
        n((x-cx).^2+(y-cy).^2<r.^2) = 1;
        model(1:size(n,1),1:size(n,2)) = n;
    case 'whole'
        r = 9;
        model = zeros(round(ceil(max(p,q)/2/(r+1))*3*(r+1)));
        siz = size(model,1);
        sx = round(siz/2);
        i = 1:round(siz/2/(r+1));
        j = 1:round(0.9*siz/2/(r+1));
        j = j-round(median(j));
        model(sx+2*j*(r+1),(2*i-1)*(r+1)) = 1;
        se = strel('disk',r);
        model = imdilate(model,se);
        model = model(round(siz/2-p/2-6):round(siz/2-p/2-6)+p-1,round(siz/2-q/2-
6):round(siz/2-q/2-6)+q-1);
end
        tem(:,:,1) = model;
        M = padarray(model,[floor(2/3*r),floor(2/3*r)],0,'post');
        tem(:,:,2) = M(floor(2/3*r)+1:end,floor(2/3*r)+1:end);
        model = tem;
end
```