

## Activity 9: Contour-based Segmentation



Primary Goal: Code the Chan-Vese (CV) segmentation algorithm.

Secondary Goal: Learn how to work with level set functions.

We can track a closed contour  $\Gamma$  using a level set function  $u$ . We define

$$u(x, y) > 0 \Rightarrow \text{Pixel } (x, y) \text{ is inside } \Gamma$$

$$u(x, y) < 0 \Rightarrow \text{Pixel } (x, y) \text{ is outside } \Gamma$$

The contour  $\Gamma$  can be located by computing where  $u = 0$ . Note that digitally  $u$  may cross zero in between pixels, you do not necessarily get zero values in the matrix  $u$ .

Let's create two example level set functions and examine their contours. Since we will want the level set to correspond to a grayscale image, let's read the "coins" image.

```
A = imread('coins.png');  
[m,n] = size(A);
```

### 1.) Circular Step Function

We generally prefer our level set functions to be smooth shapes like circles. One way to make a binary circle image with center  $(m/2, n/2)$  and radius 50 is with the `meshgrid` command.

```
[x,y] = meshgrid(1:n, 1:m);  
D = [(x - n/2).^2 + (y - m/2).^2 <= 50^2];
```

If you `imshow` this image, you will see it is a white circle (1) on a black background (0).

To make this image take on negative values outside the circle, we could make a level set function:


```
u = 2*D - 1;
```

This algebraic transformation makes the new image  $u$  take on values -1 and 1 (rather than 0 and 1).

There are four basic ways to view a level set function.

i.) We could examine the 3D surface plot.

```
surf(u)
```

Press the "Rotate 3D" button  on the figure's menu to spin the surface.

ii.) We could also display the level set function as an image.

```
imagesc(u); colorbar;
```

iii.) We can display the contour at the zero-crossings of  $u$ .

```
contour(u, [0,0], 'r')
```

iv.) If you would like to display a red contour on top of the image  $A$ , we could use the commands:

```
imshow(A); hold on;  
contour(u, [0,0], 'r'); hold off;
```

Try changing the radius of the circle to 100.

## 2.) Circular Signed Distance Function

Unfortunately, the simple level set function in #1 will cause problems numerically in the CV segmentation algorithm. While the contour is smooth, the gradient of this function is not smooth because it makes a sudden jump from 0 to 1. This sudden jump can throw off the curve evolution and lead to unexpected results.

A common trick to create a smooth level set function is to use the signed distance function. This function indicates the distance to the contour, with positive values inside and negative values outside. The Matlab command `bwdist` computes the distance of each pixel to the nearest 1 in a binary image. An example of using this function on a 6x6 binary matrix D is shown below.

D						bwdist(D)						bwdist(1-D)					
0	0	0	0	0	0	2.2	1.4	1	1	1	1.4	0	0	0	0	0	0
0	0	1	1	1	0	2	1	0	0	0	1	0	0	1	1	1	0
0	0	1	1	1	0	2	1	0	0	0	1	0	0	1	2	1	0
0	0	1	1	1	0	2	1	0	0	0	1	0	0	1	1	1	0
0	0	0	0	0	0	2.2	1.4	1	1	1	1.4	0	0	0	0	0	0
0	0	0	0	0	0	2.8	2.2	2	2	2	2.2	0	0	0	0	0	0

**Figure 1.** Illustration of the `bwdist` function.

If we use this function cleverly, we can create a signed distance from our circle image D. We first create our binary circle image D as before.

```
[x,y] = meshgrid(1:n, 1:m);
D = [(x - n/2).^2 + (y - m/2).^2 <= 100^2];
```

Next we will set u to be the signed distance to the circumference of the circle in D.

```
u = bwdist(1-D) - bwdist(D);
u = 0.1*u;
```

The second line scales the level set function down, which will make our segmentation algorithm a little faster.

View the level set function u using each of the four methods in #1. Note u takes on positive values inside the circle and negative values outside. Compared to the level set function in #1, this function makes a gradual transition between values from one pixel to the next.

### 3.) Chan-Vese Active Contours

The Chan-Vese (CV) segmentation method evolves a contour  $\Gamma$  to fit snugly around the objects in an image. The two-phase (binary) segmentation for a grayscale image  $f$  is given by minimizing the energy

$$\min_{\Gamma} E_{CV}[\Gamma|f] = L(\Gamma) + \lambda \int_{\text{inside } \Gamma} (f - c_{in})^2 d\vec{x} + \lambda \int_{\text{outside } \Gamma} (f - c_{out})^2 d\vec{x}$$

where the average gray values are given by

$$c_{in} = \frac{\int_{\text{inside } \Gamma} f d\vec{x}}{\int_{\text{inside } \Gamma} d\vec{x}} \quad c_{out} = \frac{\int_{\text{outside } \Gamma} f d\vec{x}}{\int_{\text{outside } \Gamma} d\vec{x}}.$$

To track the contour  $\Gamma$ , we examine the zero level curve of a level set function  $u$ . We will assume the level set function satisfies  $u > 0$  inside  $\Gamma$  and  $u < 0$  outside  $\Gamma$ . Rewriting the energy in terms of  $u$ , calculating the first variation of energy, and applying steepest descent means we should evolve the PDE:

$$\frac{\partial u}{\partial t} = \delta(u) \left[ \frac{u_{xx}u_y^2 - 2u_xu_yu_{xy} + u_{yy}u_x^2}{(u_x^2 + u_y^2)^{3/2} + a} - \lambda (f - c_{in})^2 + \lambda (f - c_{out})^2 \right]$$

It would help to smooth the Dirac delta by

$$\delta_{\varepsilon}(u) = \frac{\varepsilon}{\pi(\varepsilon^2 + u^2)}$$

**a.)** Your goal is to code up Chan-Vese segmentation using level sets. Create a function called `CV` that takes a grayscale image  $f$  and a  $\lambda$  value as input and returns the final level set function  $u$ :

```
function [u] = CV(f, lambda)
```

**b.)** In your function, first initialize the level set function  $u$  to be the signed distance function of a large circle of radius 100 using the commands in #2.

**c.)** Next set the parameters. For my version, I used the parameters  $\varepsilon = 0.1$ ,  $\Delta t = 0.2$ ,  $a = 0.1$ , and stopping time  $T = 100$ .

**d.)** You should update the average values  $c_{in}$  and  $c_{out}$  at the start of each iteration. You can determine which pixels are inside  $\Gamma$  by finding where  $u > 0$ . For example, you could approximate the integral  $\int_{\text{inside } \Gamma} f d\vec{x}$  with the Matlab command: `sum(sum( f.*[u>0] ));`

**e.)** Next set the update for the level set function  $u$  to evolve the PDE above. Note the first term of the PDE inside the brackets is the same as the curvature term in TV denoising. You can copy and paste code for this term from previous assignments.

**f.)** Try drawing the contour every iteration. To draw a contour  $\phi$  on top of the image  $f$ , use the Matlab commands:

```
imshow(uint8(f)); colormap gray;
hold on; contour(u, [0,0], 'r'); hold off; drawnow;
```

**g.)** To call your function, you need to send it an image and a  $\lambda$  value. Let's try  $\lambda = 0.001$ .

```
A = imread('coins.png');
u = CV (A, 0.001);
```

Next try the larger value  $\lambda = 0.1$ . What happens to the contour and why?