Assignment #3

Level-set Image Segmentation

학번: 2012210060

이름: 정 치훈

Contents

1. Assign_3_skeleton.m

(a) Source code

```
% ToDO ------------------------
%First way [Gaussian smoothing]
G = imgaussfilt(Img,1); % imgaussfilt(A, sigma) A:image
[x,y]=gradient(G);
f = x.^2+y.^2;
g = 1./(1+f);
%Second way[Gaussian smoothing]
%G=fspecial('gaussian',200,1);  [% fspecial('gaussian', hsize, sigma)]
%smooth_img=conv2(Img,G,'same');  [% smooth image by Gaussiin convolution]
%[x,y] = gradient(smooth_img);
%f = x.^2+y.^2;
%g = 1./(1+f);
% ------------------------------
```

$$g = \frac{1}{1+|\nabla \hat{I}|^p}$$

(b) explanation

$$g = \frac{1}{1+|\nabla \hat{I}|^p}$$

where $\hat{I}$ is a smoothed version of the input image I, and p = 2. You can use any image smoothing algorithm to generate $\hat{I}$ (Gaussian smoothing is commonly used). This edge indicator works as follows: when the level set curve is moving closer to edges, then the magnitude of gradient becomes larger and therefore g becomes smaller, which makes the movement of the curve smaller.

In order to update the distance field u, you need to compute the above terms and update as follows:

As following mentioned, Gaussian smoothing is commonly used to generate " $\hat{I}$ ". There are two options that compute gaussian smoothing in matlab.

I used both two options to compute gaussian smoothing. Imgaussfilt(A, sigma) function and fspecial('gaussian', hsize, sigma) function. There are both have advantages in each function that first function is easy to compute gaussian smoothing without convolution code, second function can control 'hsize' and 'sigma' in detail.

2. Levelset_update.m

(a) Source code

```
% ToDo
%---------------------------------------------------------------------------
[phi_x,phi_y] = gradient(phi_in);
norm_gra = sqrt(phi_x.^2+phi_y.^2+1e-10); % the norm of the gradient plus a small possitive number
X = phi_x./norm_gra;
Y = phi_y./norm_gra;
[xx,None] = gradient(X);
[None,yy] = gradient(Y);

dPhi = sqrt(phi_x.^2+phi_y.^2); % mag(grad(phi))

kappa = xx + yy; % curvature
%---------------------------------------------------------------------------
```

$$u' = u + dt(g(I)(c + \boxed{k})\boxed{|\nabla u|})$$

(b) Explanation

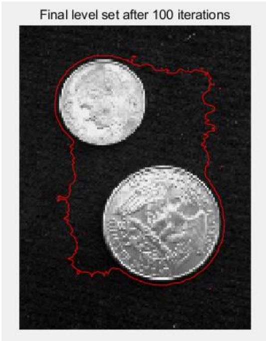
when we get the gradient of image, we can get the two values that x_grad, y_grad, and we derivate this values again, then we get 4 images xx_grad, xy_grad, yx_grad, yy_grade in finally. Kappa value is the sum of xx_grad and yy_grad.

[dphi represented a magnitude of u gradient, so the equation up below.]

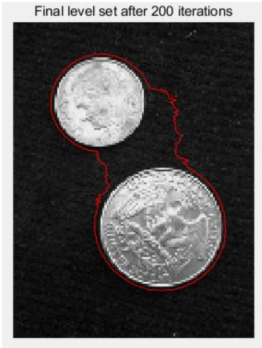
3. Result & Experiments

(a) Result image of each iterations

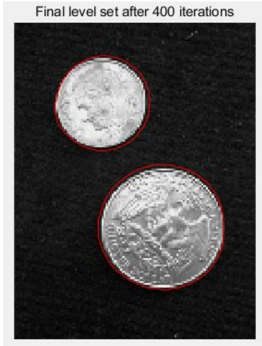
Iteration = 10



Iteration = 100



Iteration = 200

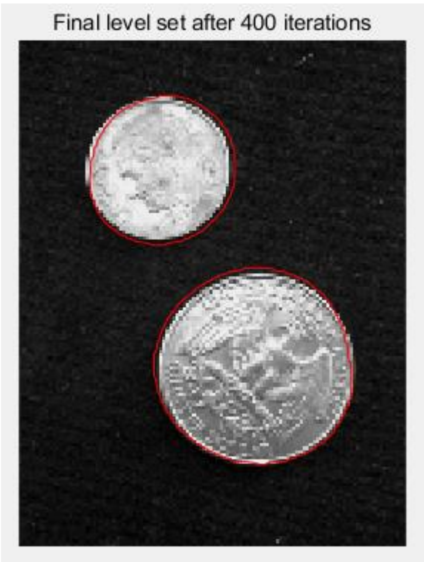


Iteration = 400

(b) Experiments of dt(time step) & sigma(gaussian)

1. Conditions: sigma = 1, using imgaussfilt() function.

When 0.9 > dt > 0.5



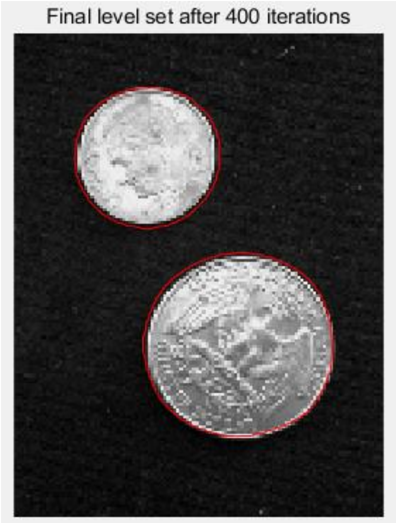
dt = 0.6

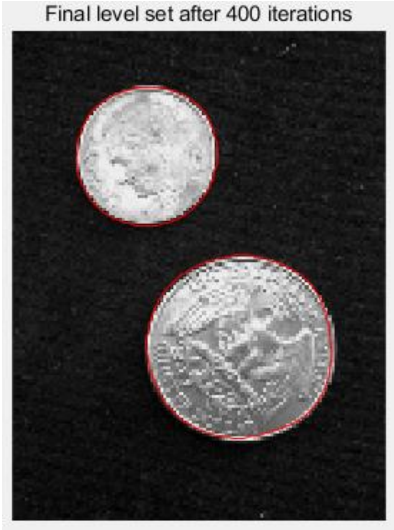
dt = 0.7

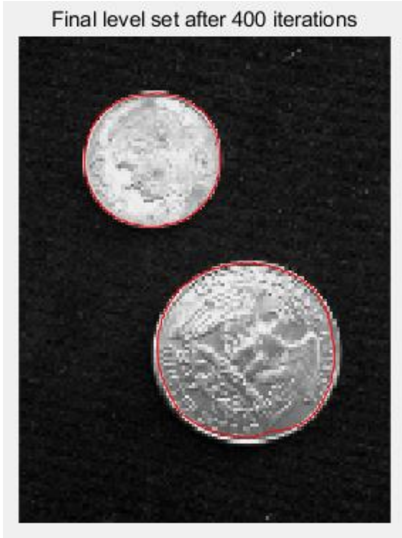
dt = 0.8

2. Conditions: sigma = 2, using imgaussfilt() function.

When 0.9 > dt > 0.5





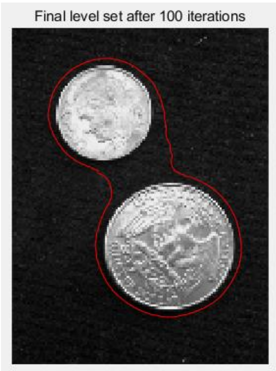


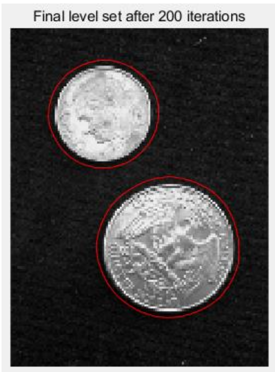
[BEST] dt = 0.6 **dt = 0.7** **dt = 0.8**

3. Conditions: sigma = 2, dt = 0.5 using imgaussfilt() function.

[Notice: compare with “Result Image”]

When iterations = 10, 100, 200, 400

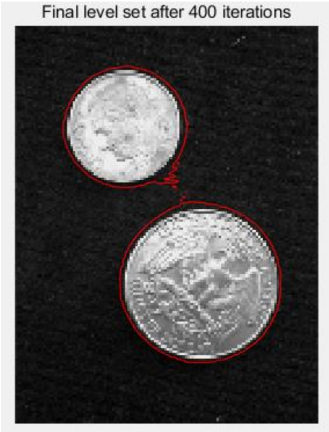




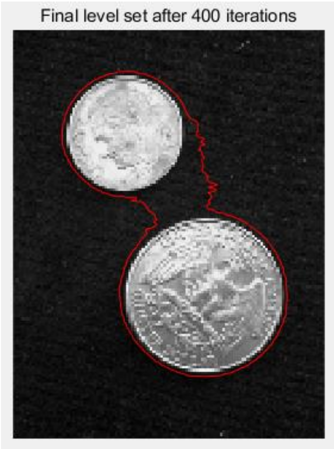
Iter= 10 **Iter= 100** **Iter= 200** **Iter= 400**

4. Conditions sigma = 1, using imgaussfilt() function.

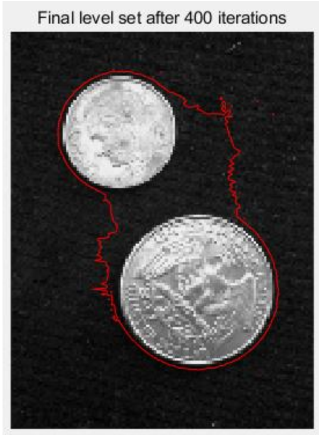
When 0.5 > dt



dt= 0.4



dt= 0.3



dt= 0.2

dt= 0.1

4. Conclusion

As following experiments, if dt is increase, the active contour is moving faster and if dt is decrease, the active contour is moving slower. Given functions define that using gradient to find edge of object. But through the lots of experiment, we can figure out bigger noise image or more smoothing image cannot get better result of active contour. Which means, as the experiments higher value of sigma then, the image is much smoothing than before it makes hard to finding edges of image.

In my opinion, among the many experiments sigma = 2 & dt = 0.6 can get best result of active contour.