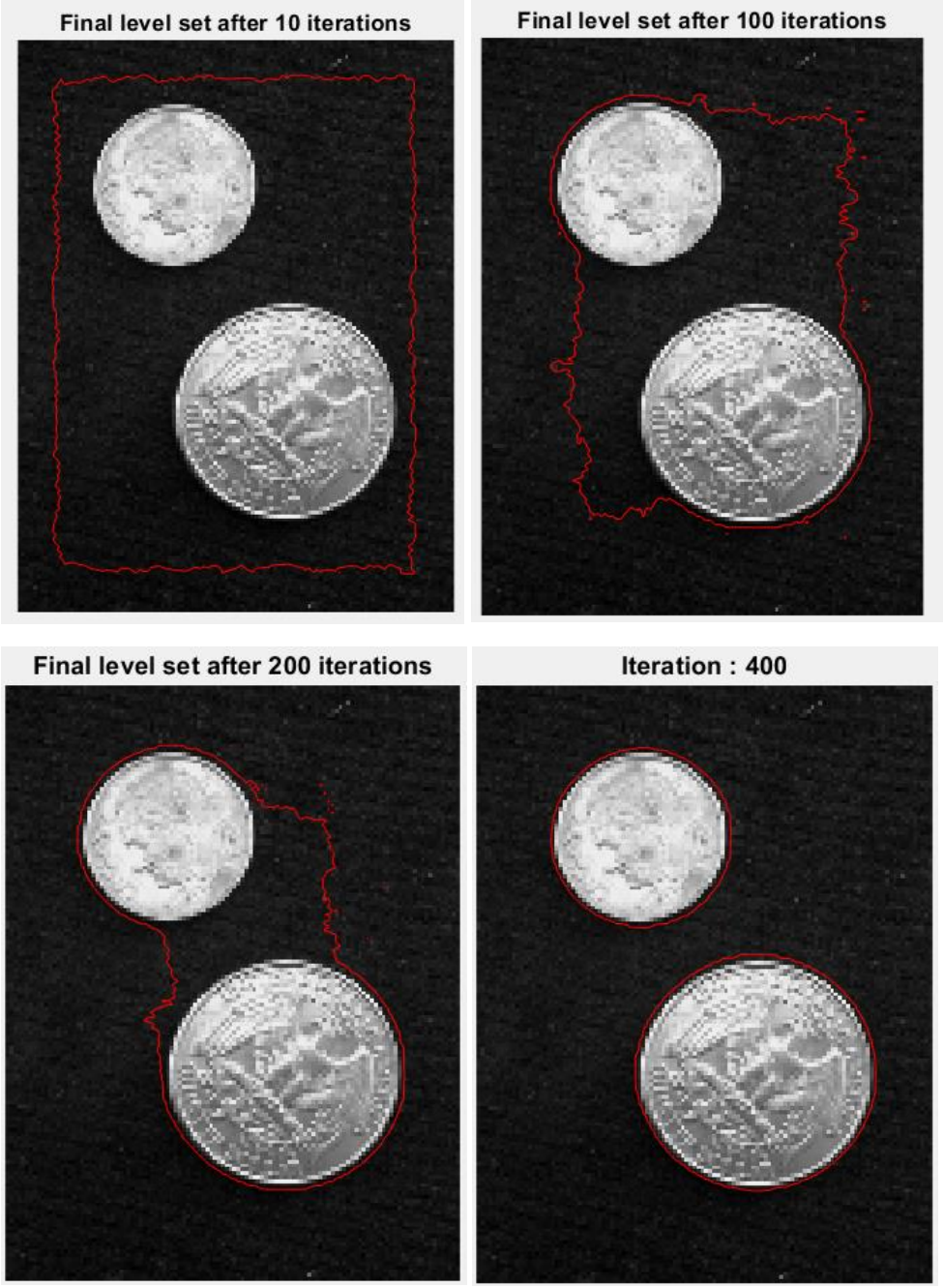
학번: 2017320215

이름: 임준상

실행 결과(10, 100, 200, 400)



assign_3_skeleton:

assign_3_skeleton 에서 해야 하는 것은 g(I)를 계산하는 것입니다.

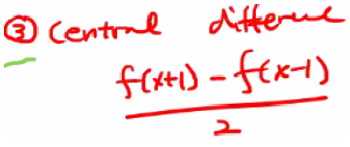
```
37      % ToDO ------------------------
38      % Gaussian smoothing
39 -    sigma = 1.5;
40 -    Gweights = fspecial('gaussian', [3,3], sigma);
41 -    I = Img;
42 -    for i = 1: numRows
43 -        for j = 1: numCols
44 -            if i~=1 && j~=1 && i~=numRows && j~=numCols
45 -                temp = Img(i-1:i+1, j-1:j+1);
46 -                I(i, j)=sum(sum(temp .* Gweights));
47 -            end
48 -        end
49 -    end
50 -    p = 2.0;
```

먼저 Gaussian smoothing 합니다. fspecial 함수를 이용해 Gaussian filter 를 얻어서 이미지를 filtering 합니다.

그 다음에 이미지의 gradient(central diff.)를 계산해서 gradient 의 magnitude 를 계산합니다. 이미지의 gradient magnitude 를 얻어서 g(I)를 계산합니다.

```
52      % gradient of I
53 -    gra_x = zeros(numRows, numCols);    % gradient of x
54 -    gra_y = zeros(numRows, numCols);    % gradient of y
55 -    temp = zeros(numRows+2, numCols+2);
56 -    temp(2:numRows+1, 2:numCols+1) = I;
57      % central diff.
58 -    for i = 2:numRows+1
59 -        for j = 2:numCols+1
60 -            gra_x(i-1, j-1) = (temp(i, j+1) - temp(i, j-1)) / 2;
61 -        end
62 -    end
63 -    for i = 2:numRows+1
64 -        for j = 2:numCols+1
65 -            gra_y(i-1, j-1) = (temp(i+1, j) - temp(i-1, j)) / 2;
66 -        end
67 -    end
68 -    gradient_I = sqrt(gra_x.^2 + gra_y.^2);
69
70 -    g = 1 ./ (1+(gradient_I.^p));
```

central diff.와 magnitude 의 계산법은 다음과 같습니다.



- Magnitude of the gradient

$$\nabla f = \left(G^2_x + G^2_y\right)^{1/2} \qquad \nabla f \approx |G_x| + |G_y|$$

levelset_update:

levelset_update 에서 gradient 를 계산하는 함수를 만들었습니다. 위와 같은 central diff.를 구현하는 gra 함수입니다.

```
31    function [grax, gray] = gra(input)
32 -  [rows,cols] = size(input);
33 -  grax = zeros(rows, cols);   % gradient of x
34 -  gray = zeros(rows, cols);   % gradient of y
35 -  temp = zeros(rows+2, cols+2);
36 -  temp(2:rows+1, 2:cols+1) = input;
37
38    % central diff
39 -  for i = 2:rows+1
40 -      for j = 2:cols+1
41 -          grax(i-1, j-1) = (temp(i, j+1) - temp(i, j-1)) / 2;
42 -      end
43 -  end
44
45 -  for i = 2:rows+1
46 -      for j = 2:cols+1
47 -          gray(i-1, j-1) = (temp(i+1, j) - temp(i-1, j)) / 2;
48 -      end
49 -  end
50
51 -  end
```

dt 가 0.8 으로 설정하면 그림은 niter 이 300 이 될 때부터 더이상 변하지 않아서 0.6 으로 설정했습니다. 0.8 으로 해야하면 8 줄을 빼시면 됩니다.

gra()함수를 이용해 dPhi 의 gradient 를 얻어서 magnitude 를 계산하고(10~11 줄) gradient 를 자기 magnitude 를 나눕니다(14~15 줄).

```
10 -  [dPhi_x, dPhi_y] = gra(phi_in);
11 -  dPhi = sqrt(dPhi_x.^2 + dPhi_y.^2 + 1.0e-8.^2);
12
13    %
14 -  dPhi_x = dPhi_x ./ dPhi;
15 -  dPhi_y = dPhi_y ./ dPhi;
```

아래는 divergence 공식입니다.

$$A(x,y,z) = P(x,y,z)i + Q(x,y,z)j + R(x,y,z)k, \qquad divA = \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial k}$$

gradient_x 가 P 으로, gradient_y 가 Q 으로 간주하면 divergence 의 값은 아래 코드로 구할 수 있습니다(17~20 줄).

```
17 -  [phix_x, temp] = gra(dPhi_x);
18 -  [temp, phiy_y] = gra(dPhi_y);
19
20 -  kappa = (phix_x + phiy_y);
```

수업에서 kappa 는 second order derivative 라고 하셨는데 이 부분 코드또 짰습니다. 다만 저는 첫번째 방법으로 실행하는 코드의 boundary 가 더 명확하다고 생각해서 첫번째 방법을 사용했습니다.

```
22      % use second derivative
23      % temp = (dPhi_x + dPhi_y) ./ dPhi;
24      %
25      % [a, b] = gra(temp);
26      %
27      % kappa = (a + b);
```

이 부분 코드를 해보고 싶어하시면 **14** 줄에서 **20** 줄까지의 코드를 주석하고 **23** 줄에서 **27** 줄까지의 코드 주석을 취소하시면 됩니다.