

**Medical Image Analysis**  
**Professor Ganapati Krishnamurti**  
**Biomedical Engineering Design**  
**Indian Institute of Technology, Madras**

**Lecture 32**

**Segmentation Models Demo**  
**[Snakes (Active Contours), Chan-Vese Segmentation, Geodesic Active Contour]**

(Refer Slide Time: 00:16)

The screenshot shows a presentation slide with the following content:

- Header: "Please recall"
- Section: "STEP-1 Definition of energy functional"
- Text: "Snakes- Euler-Lagrange Equations"
- Equation: 
$$E_{int} = \alpha \left( \frac{\partial C(s)}{\partial s} \right)^2 + \beta \left( \frac{\partial^2 C(s)}{\partial s^2} \right)^2, \alpha, \beta > 0$$
- Equation: 
$$E_{ext}(C(s)) = - || \nabla f(C(s)) ||^2$$
- Equation: 
$$\min \int_0^1 E_{int}(C(s)) + E_{ext}(C(s)) ds$$
- Text: "Here the minimum of the energy E is set to the curve C(s) or (x,y) pairs defining the curve"
- Equation: 
$$\vec{C}(s) = \begin{Bmatrix} x^*(s) \\ y^*(s) \end{Bmatrix}$$
- Text: "For ex: If  $\vec{C}(s)$  is a circle"
- Equation: 
$$\vec{C}_k = \begin{Bmatrix} x(s) \\ y(s) \end{Bmatrix} = \begin{Bmatrix} x_c + r \cos(2\pi s) \\ y_c + r \sin(2\pi s) \end{Bmatrix}$$
- Text: " $s \in [0, 1]$ ,  $(x_c, y_c)$ : Center,  $r$ : Radius"

Hello everyone. Welcome to the MATLAB tutorial of Medical Image Analysis course by Professor Ganapati Krishnamurti. In today's tutorial, we will demonstrate 3 active contour based segmentation techniques discussed by the professor in the lecture videos. These techniques are active snakes, geodesic active contours and Chan-Vese segmentation. Please, carefully go through the lecture before this tutorial. Because we will focus only on the numerical method aspects and not on the theory. So, we start with snakes.

As professor discussed, the energy functional to be minimized for snakes consists of 2 parts, a external energy or data term and b internal energy or regularization term. The expression for the 2 terms are shown on your screen. Please, note that C here represents the parameterized equation of snake. For example, the parameterized equation of a circle is shown on the right hand side. C vector is equal to  $x^*(s), y^*(s)$  and this is parameterized by s. Here, the parameter s varies between 0 to 1.

(Refer Slide Time: 01:43)

1 of 16 MIA\_Active\_contour\_Tutorial.pdf 120%

Here the minimums of the energy E is set to the curve C(s) or (x,y) points defining the curve.  $S \in [0,1]$ ,  $(x_c, y_c)$ : Center,  $r$ : Radius

STEP-2 Euler-Lagrange equations

EL -Higher Order Derivatives

$$J[y] = \int_a^b f(x, y(x), y'(x), y''(x)) dx$$

$$y(a) = y_a, \quad y(b) = y_b, \quad y'(a) = y'_a, \quad y'(b) = y'_b.$$

$$\frac{\partial f}{\partial y} - \frac{d}{dx} \left( \frac{\partial f}{\partial y'} \right) + \frac{d^2}{dx^2} \left( \frac{\partial f}{\partial y''} \right) = 0.$$

In our case,

$$J(c(s)) = \int_0^1 E_{\text{int}}(c(s)) + E_{\text{ext}}(c(s)) ds$$

1 of 16 MIA\_Active\_contour\_Tutorial.pdf 120%

EL -Higher Order Derivatives

$$J[y] = \int_a^b f(x, y(x), y'(x), y''(x)) dx$$

$$y(a) = y_a, \quad y(b) = y_b, \quad y'(a) = y'_a, \quad y'(b) = y'_b.$$

$$\frac{\partial f}{\partial y} - \frac{d}{dx} \left( \frac{\partial f}{\partial y'} \right) + \frac{d^2}{dx^2} \left( \frac{\partial f}{\partial y''} \right) = 0.$$

In our case,

$$J(c(s)) = \int_0^1 E_{\text{int}}(c(s)) + E_{\text{ext}}(c(s)) ds$$

To find the optimum  $c$ , we have to solve Euler-Lagrange equation involving second order terms. Here is the formula for it. The value of  $j$  and  $f$  here, in our case would be this.

(Refer Slide Time: 02:08)

Handwritten red equation:  $f = E_{int}(C(s)) + E_{ext}(C(s))$

### Snakes-Euler-Lagrange Equations

$$-\alpha \frac{\partial}{\partial s} \left( \frac{\partial C(s)}{\partial s} \right) + \beta \frac{\partial^2}{\partial s^2} \left( \frac{\partial^2 C(s)}{\partial s^2} \right) + \nabla E_{ext}(C(s)) = 0$$

The node positions  $C(s)$  vary with time, as the curve changes to minimise the functional. The equations hold when the optimal curve minimising the functional is found. We can now provide a gradient descent Equation for minimising the energy functional

**STEP-3 Snake evolution equation**

So, on plugging these expressions, we get Euler-Lagrange equation as shown here. Now, to find the snake evolution equation that is step 3, we convert this problem into a pseudo transient problem by introducing an unsteady term.

(Refer Slide Time: 02:33)

### Snake evolution equation

The node positions  $C(s)$  vary with time, as the curve changes to minimise the functional. The equations hold when the optimal curve minimising the functional is found. We can now provide a gradient descent Equation for minimising the energy functional

$$\frac{\partial C(s)}{\partial t} = -\delta J$$

**STEP-3 Snake evolution equation**

So, here is the unsteady term that we have introduced and now we get the following equation, following unsteady equation. So, when we do it, we get the following equation.

(Refer Slide Time: 02:51)

$$\frac{\partial \vec{C}(s)}{\partial t} = \alpha(s) \frac{\partial^2 \vec{C}(s)}{\partial s^2} - \beta(s) \frac{\partial^4 \vec{C}(s)}{\partial s^4} - \begin{bmatrix} \frac{\partial}{\partial z^x} \\ \frac{\partial}{\partial y^x} \end{bmatrix} E_{ext}(C(s))$$

(I)                      (II)                      (III)                      (IV)

STEP-4 Discretization

Here, alpha and beta are the 2 parameters that determine the elastic and bending properties of snake and here these parameters are function of f. But we assume that these are constants. So, here one approximation comes into the, our first approximation comes into picture and these equations get reduced to this one. So, here alpha is alpha, beta is beta. Now, this is the main snake evolution equation and you can see it consists of 4 terms.

So, now, we will discuss its numerical solution. Before that please, note that we are using  $(x^*, y^*)$  to represent snake coordinates and they are different from Cartesian coordinates x, y. For example, here you can see the gradient in the fourth term, here, is taken with respect to snake coordinates and not Cartesian coordinates.



(Refer Slide Time: 04:22)

$\frac{\partial}{\partial t}$        $\frac{\partial}{\partial s}$        $\frac{\partial^2}{\partial s^2}$        $\frac{\partial^4}{\partial s^4}$       ext.

(I)      (II)      (III)      (IV)

STEP-4 Discretization

We show discretization for just one component.

(I):

$$\frac{\partial x^*}{\partial t} = \frac{x_{s_j}^t - x_{s_j}^{t-1}}{\Delta t} = \frac{x_j^t - x_j^{t-1}}{\Delta t}$$

Similarly for  $y^*$

(II)

$$\frac{\partial^2 x^*}{\partial s^2} = \frac{x_{j+1}^t - 2x_j^t + x_{j-1}^t}{\Delta s^2}$$

NPTEL

Now, when it comes to discretization, we will show discretization only for one component but the second component will also be same, will be similar. So, for example, we start with the unsteady term. So, the unsteady term is approximated with the help of backward difference formula, which you can see here.  $x_{s_j}^t, x_{s_j}^{t-1}$ , here it is  $t$ , it is  $t - 1$ . For simplification, in shorthand, we can write  $x_{s_j}^t$  as  $x_j^t$ . So,  $x_j^t$  at  $t$  minus  $x_j^{t-1}$  upon  $\Delta t$ . So, this is the backward difference approximation of unsteady term. The second and also you have to note that, okay.

(Refer Slide Time: 05:16)

$$\frac{\partial x^*}{\partial t} = \frac{x_{s_j}^t - x_{s_j}^{t-1}}{\Delta t} = \frac{x_j^t - x_j^{t-1}}{\Delta t}$$

Similarly for  $y^*$

(II)

$$\frac{\partial^2 x^*}{\partial s^2} = \frac{x_{j+1}^t - 2x_j^t + x_{j-1}^t}{\Delta s^2}$$

(III)

$$\frac{\partial^4 x^*}{\partial s^4} = \frac{x_{j+2}^t - 4x_{j+1}^t + 6x_j^t - 4x_{j-1}^t + x_{j-2}^t}{\Delta s^4}$$

(IV)

NPTEL

Now, we come to the second and fourth derivatives, these 2 terms. The second and the fourth derivatives are calculated using central difference at present time. You see we could have calculated these derivatives at  $t - 1$  as well but we are calculating it at present time  $t$ . So, this is common. When we solve a partial differential equation using fully implicit methods then what we do? We calculate spatial derivatives at present time. So, it looks like, this is implicit kind of thing.

(Refer Slide Time: 05:55)

(IV)

Semi-Implicit Method

Please note that (II) and (III) terms are computed  
at  $t = t$ , but (IV) term is computed at  $t = t - 1$ .

Implicit

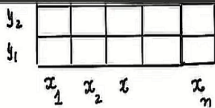
$y_n$				
$y$				
$y_2$			$I(x, y)$	
$y_1$				

Explicit

Now, we come to the last term. Although, the second and fourth terms were computed at present time like we do in implicit method, the fourth term is computed at past time like in explicit method that is at  $t$  is equal to  $t - 1$ . So, because the fourth term is calculated at past time and the other terms are calculated at present time, so, that is why it is called semi-implicit method. Now, I explained the discretization of this term.

(Refer Slide Time: 06:38)

4 | of 16 MIA\_Active\_contour\_Tutorial.pdf 120%



Given  $I(x, y)$ ,  $\nabla I(x, y)$  can be computed as follows

$$\frac{\partial I}{\partial x} = \frac{I(x_{i+1}, y_j) - I(x_{i-1}, y_j)}{2h}$$

$$\frac{\partial I}{\partial y} = \frac{I(x_i, y_{j+1}) - I(x_i, y_{j-1})}{2h}$$

$$E_{\text{ext}} = -|\nabla I|$$

$$\frac{\partial E_{\text{ext}}}{\partial x^{*t-1}} = \frac{E_{\text{ext}}(x_{i+1}^{*t-1}, y_j^{*t-1}) - E_{\text{ext}}(x_{i-1}^{*t-1}, y_j^{*t-1})}{2\Delta x^*}$$

NPTEL

4 | of 16 MIA\_Active\_contour\_Tutorial.pdf 120%

Given  $I(x, y)$ ,  $\nabla I(x, y)$  can be computed as follows

$$\frac{\partial I}{\partial x} = \frac{I(x_{i+1}, y_j) - I(x_{i-1}, y_j)}{2h}$$

$$\frac{\partial I}{\partial y} = \frac{I(x_i, y_{j+1}) - I(x_i, y_{j-1})}{2h}$$

$$E_{\text{ext}} = -|\nabla I|$$

$$\frac{\partial E_{\text{ext}}}{\partial x^{*t-1}} = \frac{E_{\text{ext}}(x_{i+1}^{*t-1}, y_j^{*t-1}) - E_{\text{ext}}(x_{i-1}^{*t-1}, y_j^{*t-1})}{2\Delta x^*}$$

Similarly for  $y^*$

NPTEL

So, given an image  $I$  as a function of  $x$  and  $y$ ,  $\text{grad } I$  can be found using Cartesian coordinates and its magnitude, negative of its magnitude gives external energy. Next, we take gradient of this energy and find its component along the snakes. Along the snakes is important, so, we are trying to find its derivative, its gradient with respect to  $x^{*t-1}$ . So, the explicit term is here. So, this is how we do it. So, this is a central difference only.

(Refer Slide Time: 07:28)

Substituting,

$$\begin{aligned}
 & b x_{j+2}^{*t} - (a+4b) x_{j+1}^{*t} + (1+2a+6b) x_j^{*t} - (a+4b) x_{j-1}^{*t} + b x_{j-2}^{*t} \\
 & = x_j^{*t-1} + \Delta t \left. \frac{\partial E_{ext}}{\partial x^*} \right|_{t-1}
 \end{aligned}$$

Where  $a = \alpha \Delta t / \Delta s^2$   
 $b = \beta \Delta t / \Delta s^4$

In vector form,

$$\begin{aligned}
 & b x_{j+2}^{*t} - (a+4b) x_{j+1}^{*t} + (1+2a+6b) x_j^{*t} - (a+4b) x_{j-1}^{*t} + b x_{j-2}^{*t} \\
 & = x_j^{*t-1} + \Delta t \left. \frac{\partial E_{ext}}{\partial x^*} \right|_{t-1}
 \end{aligned}$$

Where  $a = \alpha \Delta t / \Delta s^2$   
 $b = \beta \Delta t / \Delta s^4$

In vector form,

$$M \underline{x}^{*t} = \underline{x}^{*t-1} + \Delta t \left. \frac{\partial E_{ext}}{\partial x^*} \right|_{t-1}$$

↑  
 This matrix is cyclic pentadiagonal.

Similarly, we do for y. So, when we substitute these values, we get a pentadiagonal matrix structure with coefficients as shown here.  $b$ ,  $a + 4b$ ,  $1 + 2a + 6b$ , where the value of  $a$  and  $b$  is  $\alpha \Delta t / \Delta s^2$ ,  $\beta \Delta t / \Delta s^4$ . So, in matrix form, it reads:

$$M x^{*t} = x^{*t-1} + \Delta t \frac{\partial E_{ext}}{\partial x} * |t - 1$$

Here  $M$  is the cyclic pentadiagonal matrix and we have to efficiently calculate its inverse to update the position of snake points.

(Refer Slide Time: 08:27)

5 | of 16

MIA\_Active\_contour\_Tutorial.pdf 120%

In vector form,

$$\underline{M} \underline{x}^t = \underline{x}^{t-1} + \Delta t \left. \frac{\partial E}{\partial \underline{x}} \right|_{t-1}$$

↑

This matrix is cyclic pentadiagonal.

$$\underline{x}^t = \underline{M}^{-1} \left( \underline{x}^{t-1} + \Delta t \left. \frac{\partial E}{\partial \underline{x}} \right|_{t-1} \right)$$

Similarly for  $\underline{y}^t$

NPTEL

For example, you can see here. Given  $x^{*t-1}$  and this  $\partial E / \partial x^*$ , they can find  $x^{*t}$  by this expression. But this involves calculation of  $M^{-1}$ . So, because it is a pentadiagonal system, this we can use LU factorization to calculate its inverse efficiently.

(Refer Slide Time: 08:58)

6 | of 16

MIA\_Active\_contour\_Tutorial.pdf 120%

In the lecture,

### Gradient Descent

$Ax + \frac{\partial E(C(x))}{\partial x} = 0$  The node positions  $C(x)$  vary with time, as the curve changes to minimize the functional. The equations hold when the optimal curve minimizing the functional is found. We can now provide a gradient descent equation for minimizing the energy functional.

$Ay + \frac{\partial E(C(y))}{\partial y} = 0$   $\frac{\partial C(x)}{\partial t} = -\delta I$

$x^{t+1} = (A + \gamma I)^{-1} \left( x^t - \frac{\partial E(x^t)}{\partial x} \right)$   $- \gamma (x^{t+1} - x^t) = \left( A x^{t+1} + \frac{\partial E(C(x^t))}{\partial x} \right)$

$y^{t+1} = (A + \gamma I)^{-1} \left( y^t - \frac{\partial E(y^t)}{\partial y} \right)$   $- \gamma (y^{t+1} - y^t) = \left( A y^{t+1} + \frac{\partial E(C(y^t))}{\partial y} \right)$

$\underline{M} = \underline{A} + \gamma \underline{I}$

NPTEL

So, in the lecture, this matrix M is same as  $A + \gamma I$ . So, this is just a change of notation but it is the same as that. So, now, this is the basic snake algorithm that you will find at various places online. There are many advanced versions of snakes but we are not going to discuss them here.

(Refer Slide Time: 09:32)

7 | of 16 MIA\_Active\_Contour\_Tutorial.pdf 120%

### Some MATLAB Snippets


#### A. External forces

```
% Computing external forces
[grady,gradx] = gradient(Image);
E_ext = -1 * sqrt ((gradx .* gradx + grady .* grady));
[fx, fy] = gradient(E_ext);
```

#### B. Pentadiagonal structure

```
%initializing the snake
xs=xs';
ys=ys';
[m n] = size(xs);
[mm nn] = size(fx);

%populating the penta diagonal matrix
A = zeros(m,m);
b = [(2*alpha + 6 *beta) -(alpha + 4*beta) beta];
brow = zeros(1,m);
brow(1,1:3) = brow(1,1:3) + b;
brow(1,m-1:m) = brow(1,m-1:m) + [beta -(alpha + 4*beta)]; % populating a template row
for i=1:n
    A(i,:) = brow;
    brow = circshift(brow',1)'; % Template row being rotated to egenrate different rows in
```



7 | of 16 MIA\_Active\_Contour\_Tutorial.pdf 120%

```
[mm nn] = size(fx);

%populating the penta diagonal matrix
A = zeros(m,m);
b = [(2*alpha + 6 *beta) -(alpha + 4*beta) beta];
brow = zeros(1,m);
brow(1,1:3) = brow(1,1:3) + b;
brow(1,m-1:m) = brow(1,m-1:m) + [beta -(alpha + 4*beta)]; % populating a template row
for i=1:n
    A(i,:) = brow;
    brow = circshift(brow',1)'; % Template row being rotated to egenrate different rows in
pentadiagonal matrix
end


[L U] = lu(A + gamma * eye(m,m));
Ainv = inv(U) * inv(L); % Computing Ainv using LU factorization
```

#### C. Snake evolution equation

```
%moving the snake in each iteration
for i=1:N;

    ssx = gamma*xs - kappa*interp2(fx,xs,ys);
    ssy = gamma*ys - kappa*interp2(fy,xs,ys);

    %calculating the new position of snake
    xs = Ainv * ssx;
    ys = Ainv * ssy;
```



3 of 16 MIA\_Active\_Contour\_Tutorial.pdf 120%

$$\frac{\partial \vec{C}(s)}{\partial t} = \alpha(s) \frac{\partial^2 \vec{C}(s)}{\partial s^2} - \beta(s) \frac{\partial^4 \vec{C}(s)}{\partial s^4} - \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} E_{\text{ext}}(C(s))$$

$$\frac{\partial \vec{C}(s)}{\partial t} = \alpha \frac{\partial^2 \vec{C}(s)}{\partial s^2} - \beta \frac{\partial^4 \vec{C}(s)}{\partial s^4} - \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} E_{\text{ext}}(C(s))$$

(I) (II) (III) (IV)

STEP-4 Discretization

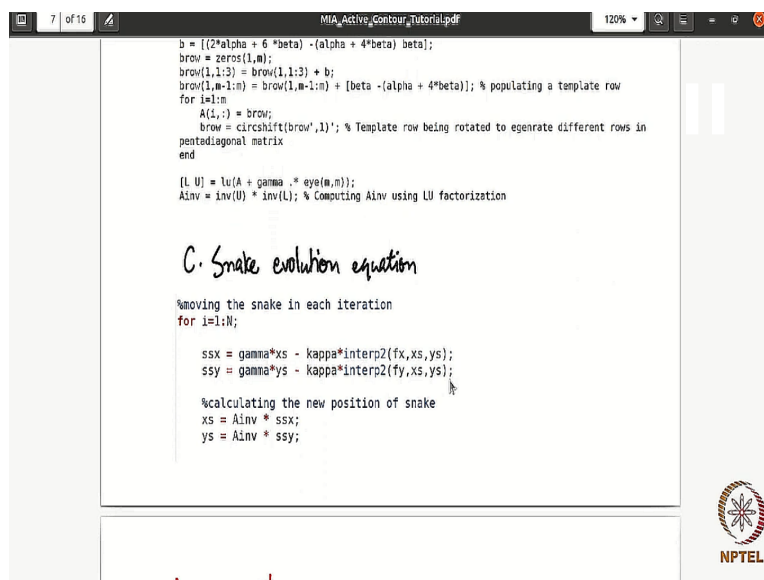
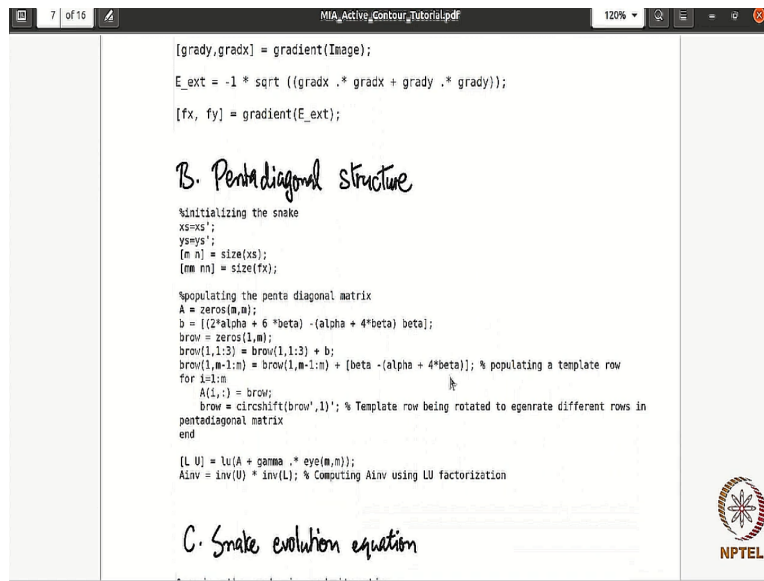
NPTEL

In fact, in the MATLAB code that we will give to you, the code has some additional forces as well. For example, balloon forces as well as the gradient vector field forces. So, there are many functions written for that purpose. Now, we will show you some basic MATLAB, some MATLAB snippets to show that the basic structure taught in the class is present in the advanced code, in the advanced code as well.

For example, here, is the code for calculating external forces that is gradient of image. Note that here, so, we can simply use gradient function in MATLAB and we can calculate energy sqrt means a square root. So, note that here  $[f_x, f_y]$ . We got  $f_x, f_y$  with the help of gradient and this  $f_x, f_y$  that we have got, it will be interpolated along the snake curves later.

For example, here, we have  $f_x, f_y$ , later we will use like we have to interpolate this  $f_x$  and  $f_y$  along  $x_s$  and  $y_s$ . These are the snake coordinates, you remember. Because we had this  $\partial/\partial x$  \* term. So, we have to do that.

(Refer Slide Time: 11:16)



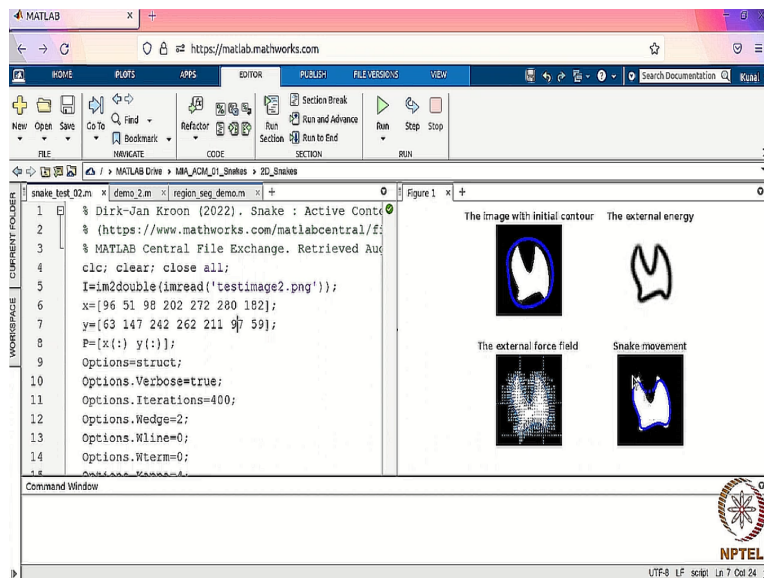
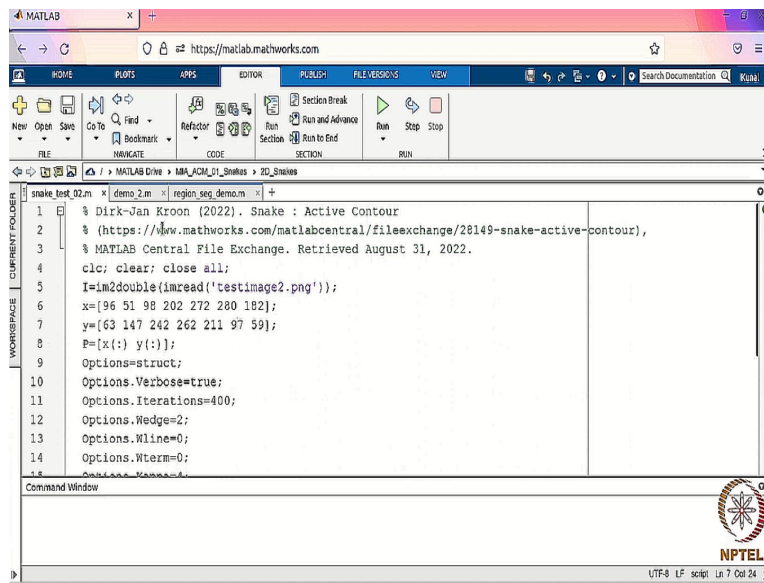
Also, this is the code for pentadiagonal structure and yeah and this is the snake evolution equation. So, thing is there are many implementations of snakes that are available online. Our objective is just to tell you that you will find the basic steps that you learned in the class, that you will find them sitting even in the advanced codes. You will have some additional forces, your functional may be different but these basic steps you will find there also.

So, when you look at a code, do not worry about it. Because it may have some additional terms, no problem. But the basic structure will be there. You will be able to see a better diagonal system. You will be like seeing people calculate some external forces. Now, what is the external force?



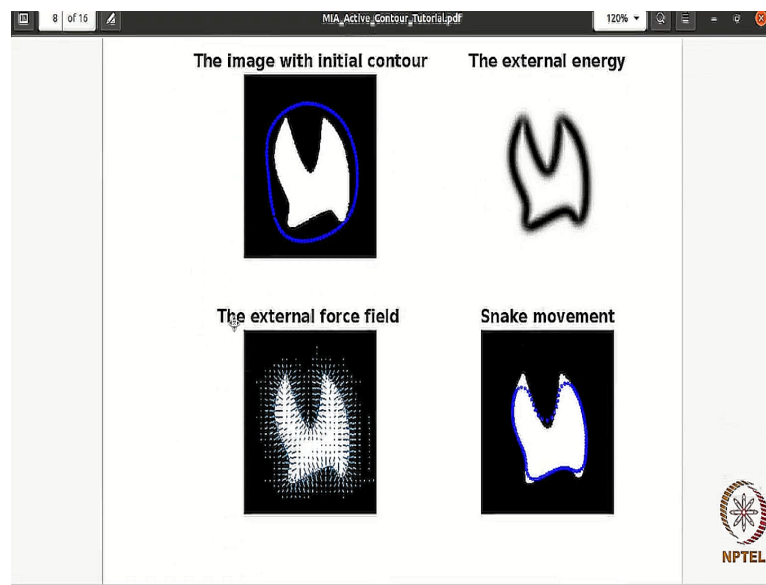
This depends on their energy functional but it will be there. So, you will use some gradient, etcetera, so, not a problem. So, even if you have one code, one MATLAB code, you can like modify it to suit your purpose. So, now, we will show you an example of snakes in action.

(Refer Slide Time: 12:31)



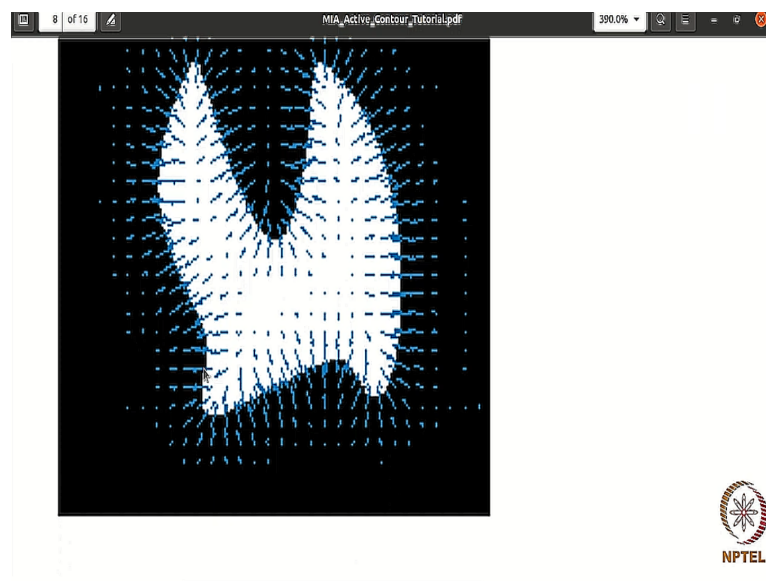
So, for example, I run this code. So, this code is not written by me, this is written by Mr. Kroon. It is available in MATLAB central. So, here you can see, this is the initial contour and you can see, now this initial contour is moving and slowly it is trying to fit this object, the boundaries of this object. So, that is how these contours move. Here, you can also see the external energy and here is external force field. So, because it is little bit slow but you have already seen that you have an initial contour that is deforming in such a way that it fits the boundary.

(Refer Slide Time: 13:39)



So, it actually looks like this. So, here you can see, the top left figure shows the boundaries to be captured and the initial contour. The right left force, the right left figure shows you the magnitude of gradient of the image. Note that gradient is high near the edge and this edge looks little bit diffused because it is convolved with the, it is convolved with a Gaussian filter. Now, in the bottom, left you see the image forces, image force vectors. So, they are small but you can magnify it. Maybe I will show you.

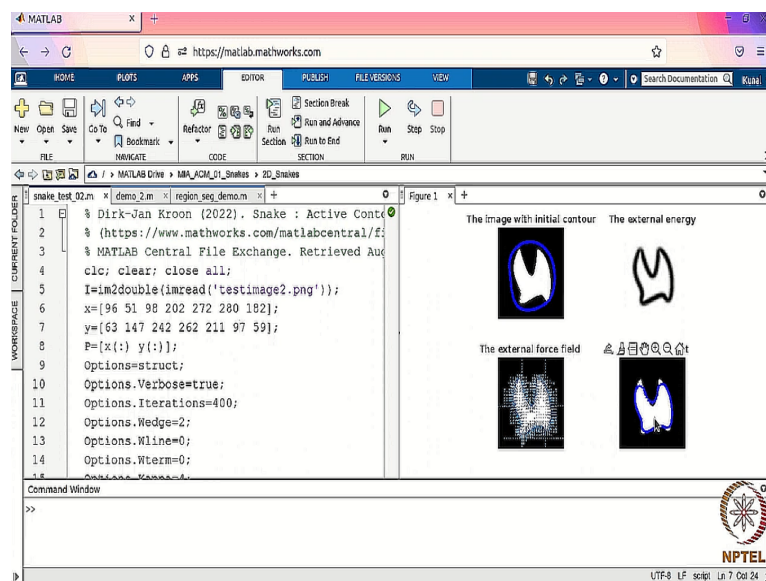
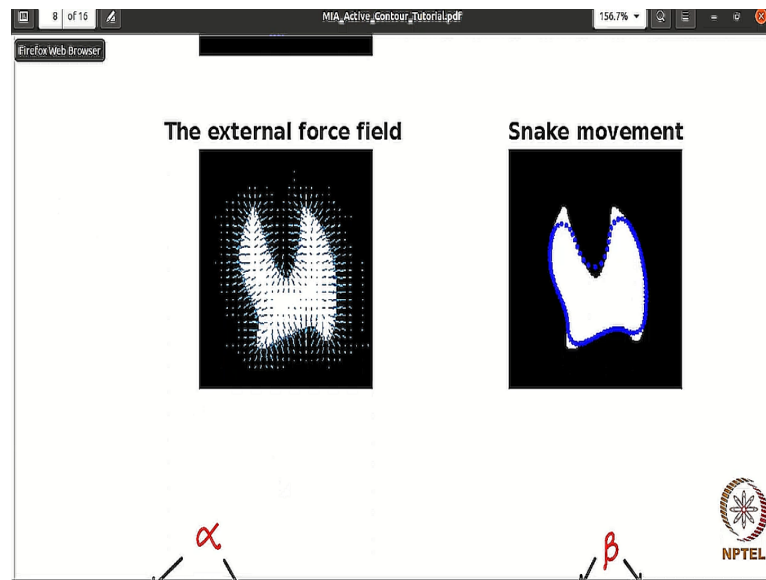
(Refer Slide Time: 14:32)



So, if you will magnify. So, they are not very clear but you can see, you have image force vectors that actually pull the contour towards the edge and their value is very high near the

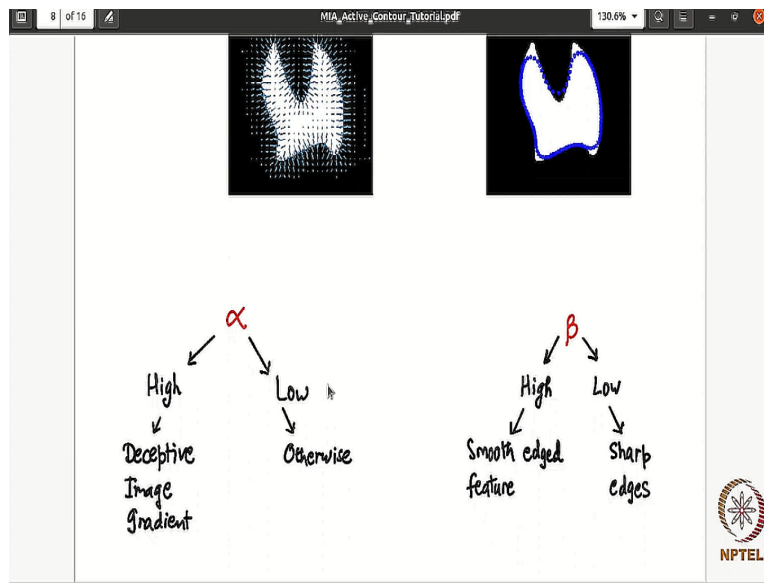
edge. As you move away from the edge, their length becomes very very less and after sometimes it becomes zero. For example, here it is all zero, here it is very less, near that, it is little bit longer.

(Refer Slide Time: 15:14)



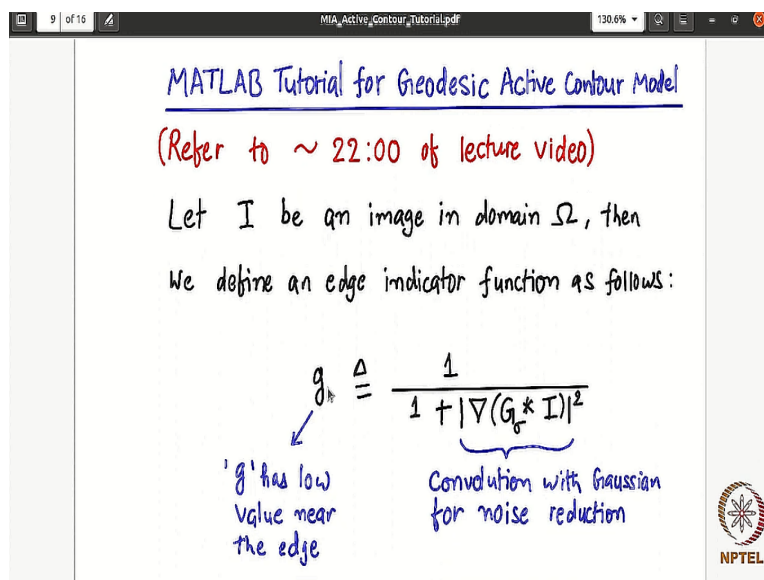
Also, the bottom, this, the bottom right figure shows the final position of the snake and as you can see the snake has fitted the object reasonably well, hopefully got it. So, you can see here. Now, that is complete. So, initially it was like this. Now, it has come to this position. So, what we have seen is that the snake has fitted the object reasonably well. So, we encourage you to play with various images, various initializations. Initialization means a different initial contouring position. And you can also play with parameters alpha, beta.

(Refer Slide Time: 16:01)



And there are some guidelines for the choice of alpha and beta. So, typically a high value of alpha is used when like your image has a deceptive image gradients and the low value of beta is used for sharp edges. So, this completes snakes, our first active contour model.

(Refer Slide Time: 16:30)



So, after this, we come to the next model, which is a Geodesic Active Contours. So, in snakes, we were explicitly involved like evolving the position of snake curve like snake points, snake coordinates by tracking position of its points. We had some points initially and we were tracking its position. So, in this approach we use level sets. So, in level set formulation what we do?

Instead of like moving the tracker points or contour points explicitly, we evolve the surface,  $\phi$ , we evolved surface  $\phi$  instead of like evolving the curve  $C$ . Earlier we were evolving the curve, curve  $C$ . But here in level formulation, we want to evolve a surface  $\phi$ . Now, the position of  $C$  is calculated implicitly, not directly. So, how do we calculate it? So,  $C$  is basically a set of points for which the value of  $\phi$  is zero or the set of points with zero height.

So, for details, I suggest you to revisit the lecture video by the professor. So, and now, I will be little fast. So, like in the case of snake, we will define an energy functional that attracts the contours of zero height that is  $C$  towards the edges. So, how do we do it? Given an image  $I$ , a simple edge indicator function is shown on your screen, it is small  $g$ . We can use this function to determine the energy functional and then solve the Euler Lagrange equation to arrive at the gradient descent equation.

(Refer Slide Time: 18:31)

'g' has low value near the edge

Convolution with Gaussian for noise reduction

(Refer to ~ 25:00 of lecture video)

For a level set formulation  $\phi: \Omega \rightarrow \mathbb{R}$   
the energy functional is given by

$$E(\phi) \triangleq \int_{\Omega} g \delta_{\epsilon}(\phi) |\nabla \phi| d\vec{x}$$

Smooth Dirac delta function

NPTEL

So, this is our energy functional. Now, it is written in terms of  $\phi$  not in terms of  $C$ . So, it is here. Here,  $\delta$  is the smooth Dirac delta function. So, I have added timestamps also. So, please, please, go to these timestamps to understand more about the theory. I am just trying to explain you the mathematical formulation.

(Refer Slide Time: 19:05)

9 | of 16 MIA\_Active\_Contour\_Tutorial.pdf 130.6%

$$g \triangleq \frac{1}{1 + |\nabla(G_\epsilon * I)|^2}$$

↓  
'g' has low value near the edge

Convolution with Gaussian for noise reduction

(Refer to ~ 25:00 of lecture video)

For a level set formulation  $\phi : \Omega \rightarrow \mathbb{R}$   
the energy functional is given by

$$E(\phi) \triangleq \int g \delta_\epsilon(\phi) |\nabla \phi| d\vec{x}$$

NPTEL

9 | of 16 MIA\_Active\_Contour\_Tutorial.pdf 130.6%

For a level set formulation  $\phi : \Omega \rightarrow \mathbb{R}$   
the energy functional is given by

$$E(\phi) \triangleq \int_{\Omega} g \delta_\epsilon(\phi) |\nabla \phi| d\vec{x}$$

↓  
Smooth Dirac delta function

NPTEL

(Refer to ~ 28:00 of lecture video)


This energy functional can be minimized by solving the following

$$\frac{\partial \phi}{\partial t} = g |\nabla \phi| \kappa + \nabla g \cdot \nabla \phi$$

$\downarrow$   
 $\nabla \cdot \left( \frac{\nabla \phi}{|\nabla \phi|} \right)$

---

In the MATLAB code shared with you, the idea is



So, this is what we do. So, we have this kind of driver force the as edge attractor. This is our energy functional and when we solve Euler-Lagrange equation, we get this level set evolution equation.

(Refer Slide Time: 19:29)


In the MATLAB code shared with you, the idea is same but functional has some additional terms. Nothing to worry about it. Here are some snapshots for your reference.

```
% Author: Chunming Li, all rights reserved
% E-mail: lichunming@gmail.com
%      li_chunming@hotmail.com
% URL: http://www.imagecomputing.org/~cmli/
```

Initial  $\phi$

$$\phi_0(x) = \begin{cases} -c_0, & \text{if } x \in R_0 \\ c_0, & \text{otherwise} \end{cases}$$

```
% initialize LSF as binary step function
c0=2;
initialLSF=c0*ones(size(img));
% generate the initial region R0 as a rectangle
initialLSF(16:55, 18:75)=-c0;
phi=initialLSF;
```



So, in the MATLAB code shared with you, the idea is same but the functional has some additional terms. Like in the case of snake, that code had some additional forces balloon forces and gradient vector field forces, so, here also it has some additional functional terms. Some additional terms in the function. But do not worry about it. So, I will show you that even though it looks complicated, the basic things that we have just saw, that we have just seen are sitting there.



So, here are some snapshots for your reference. So, for example, our initial  $\phi$ , initial level set is suppose a binary function. So, if you have to represent this, you have to write like this. In MATLAB, you can write a binary step function in this way. So, its value is either minus 2 or plus 2.

(Refer Slide Time: 20:24)

11 | of 16

MIA\_Active\_contour\_Tutorial.pdf

130.6%

$$\phi_0(x) = \begin{cases} -c_0, & \text{if } x \in R_0 \\ c_0, & \text{otherwise} \end{cases}$$

```
initialSF=c0*ones(size(img));
% generate the initial region R0 as a rectangle
initialSF(10:55, 10:75)=-c0;
phi=initialSF;
```

Edge Indicator function

$$g \triangleq \frac{1}{1 + |\nabla G_\sigma * I|^2}$$

```
G=fspecial('gaussian',15,sigma);
img_smooth=conv2(img,6,'same'); % smooth image by Gaussian convolution
[Ix,Iy]=gradient(img_smooth);
f=Ix.^2+Iy.^2;
g=1./(1+f); % edge indicator function.
```

NPTEL

9 | of 16

MIA\_Active\_contour\_Tutorial.pdf

130.6%

The edge

(Refer to ~ 25:00 of lecture video)

For a level set formulation  $\phi : \Omega \rightarrow \mathbb{R}$   
the energy functional is given by

$$E(\phi) \triangleq \int_{\Omega} g \delta_{\epsilon}(\phi) |\nabla \phi| d\vec{x}$$

↓  
Smooth Dirac delta function

NPTEL

Then the edge indicator that we have seen they have also used the same edge indicator. So, this is how we write energy indicator in MATLAB. Also, here, please, go to this almost at 25 minutes. Sir has explained that this is a smooth Dirac delta function. So, for smooth Dirac delta function, sir has written 1 upon pi into epsilon upon epsilon square plus something. It was written like this. But here, they have used some different formulation.

(Refer Slide Time: 21:10)

11 | of 16 MIA\_Active\_Contour\_Tutorial.pdf 130.6%

### Edge Indicator function

$$g = \frac{1}{1 + |\nabla G_\sigma * I|^2}$$

```
G=special('gaussian',15,sigma);
img_smooth=conv2(img,G,'same'); % smooth image by Gaussian convolution
[Ix,Iy]=gradient(img_smooth);
f=Ix.^2+Iy.^2;
g=1./(1+f); % edge indicator function.
```

### Smoothing function

$$\delta_\varepsilon(x) = \begin{cases} \frac{1}{2\varepsilon} \left[ 1 + \cos\left(\frac{\pi x}{\varepsilon}\right) \right], & |x| \leq \varepsilon \\ 0, & |x| > \varepsilon \end{cases}$$

```
function f = Dirac(x, sigma)
f=(1/2/sigma)*(1+cos(pi*x/sigma));
b = (x<=sigma) & (x>=-sigma);
f = f.*b;
```

### Level set evolution

$$\frac{\partial \phi}{\partial t} = \mu \operatorname{div} (d_p(|\nabla \phi|) \nabla \phi)$$

NPTEL

So, they are using this, this function to smooth it. You can use sir's formula also. No problem.

(Refer Slide Time: 21:17)

11 | of 16 MIA\_Active\_Contour\_Tutorial.pdf 130.6%

### Smoothing function

$$\delta_\varepsilon(x) = \begin{cases} \frac{1}{2\varepsilon} \left[ 1 + \cos\left(\frac{\pi x}{\varepsilon}\right) \right], & |x| \leq \varepsilon \\ 0, & |x| > \varepsilon \end{cases}$$

```
function f = Dirac(x, sigma)
f=(1/2/sigma)*(1+cos(pi*x/sigma));
b = (x<=sigma) & (x>=-sigma);
f = f.*b;
```

### Level set evolution

$$\frac{\partial \phi}{\partial t} = \mu \operatorname{div} (d_p(|\nabla \phi|) \nabla \phi) + \lambda \delta_\varepsilon(\phi) \operatorname{div} \left( g \frac{\nabla \phi}{|\nabla \phi|} \right) + \alpha g \delta_\varepsilon(\phi)$$

```
phi=phi_0;
[ux, uy]=gradient(g);
for i=1:iter
    phi=levelsetBoundCond(phi);
    [phi_x, phi_y]=gradient(phi);
    s=sqrt(phi_x.^2 + phi_y.^2);
```

NPTEL

11 of 16 MIA\_Active\_Contour\_Tutorial.pdf 130.6%

$$\frac{\partial \phi}{\partial t} = \mu \operatorname{div} (d_p (|\nabla \phi|) \nabla \phi) + \lambda \delta_\epsilon(\phi) \operatorname{div} \left( g \frac{\nabla \phi}{|\nabla \phi|} \right) + \alpha g \delta_\epsilon(\phi)$$

```

phi=phi_g;
[vx,vy]=gradient(g);
for k=1:iter
    phi=denoiseBoundConv(phi);
    [phi_x,phi_y]=gradient(phi);
    ssqrt1=sqrt(phi_x.^2 + phi_y.^2);
    smallNumber=1e-10;
    W=phi_x./(v+smallNumber); % add a small positive number to avoid division by zero
    W=phi_y./(v+smallNumber);
    curvature=div([Wx,Wy]);
    if strcmp(potentialFunction,'single-well')
        distRegTerm = 4*del2(phi)-curvature; % compute distance regularization term in equation (13) with the single-well
    elseif strcmp(potentialFunction,'double-well')
        distRegTerm=distReg_phi(phi); % compute the distance regularization term in equation (13) with the double-well pote
    else
        disp('Error: Wrong choice of potential function. Please input the string "single-well" or "double-well" in the crisis
    end
    diracPhi=dirac(phi,epsilon);
    areaTerm=diracPhi.*g; % balloon/pressure force
    edgeTerm=diracPhi.*(vx.*Wx+vy.*Wy) + diracPhi.*g.*curvature;
    phi=phi + timeStep*(mu*distRegTerm + lambda*edgeTerm + alpha*areaTerm);
end

```

NPTEL

Then the level set evolution equation is, it has some additional terms. This, this alpha, this mu. So, you can see this is the equation. So, you can like maybe put lambda equal to zero or you can simply turn off some of the forces and you will get a simpler model also and you can check whether it is performing well or not. So, this is the like the MATLAB code. Now, we show you a sample implementation of this code. So, let me show you.

(Refer Slide Time: 21:56)

MATLAB x +

https://matlab.mathworks.com

HOME PLOTS APPS FIGURE

Save As Show Code Line Color Line Style

FILE TEXT STYLE LINE STYLE TOOLS

Figure 1 x Figure 2 x Figure 3 x

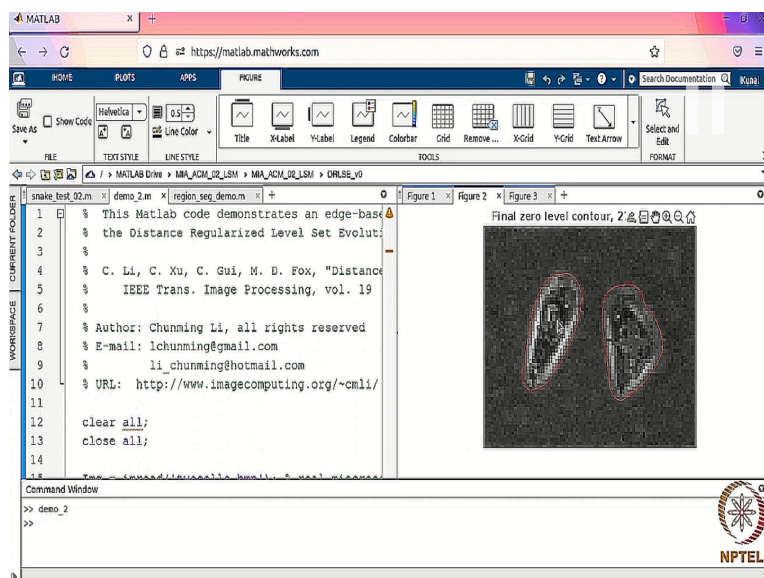
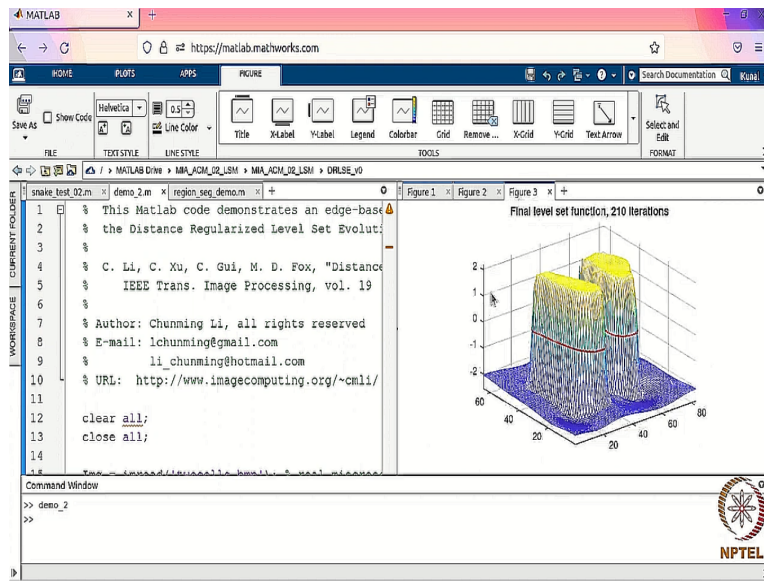
Initial level set function

1 % This Matlab code demonstrates an edge-based  
2 % the Distance Regularized Level Set Evolu  
3 %  
4 % C. Li, C. Xu, C. Gui, M. D. Fox, "Distance  
5 % IEEE Trans. Image Processing, vol. 19  
6 %  
7 % Author: Chunming Li, all rights reserved  
8 % E-mail: lchunming@gmail.com  
9 % li\_chunming@hotmail.com  
10 % URL: http://www.imagecomputing.org/~cml/  
11  
12 clear all;  
13 close all;  
14

Command Window

>> demo\_2  
>>

NPTEL



This is very very slow. So, you see, hope it will take some time. So, you see the initial level set function was this, minus 2 plus 2 the binary step function and as after some time, after some iteration because we are solving that unsteady equation, the shape of this initial level set function changes and finally, it becomes like this.

And if you see, this is the zero point, so, when the height is zero, the contour at that point is the curve  $C$ . So, if you see, the contour actually coincides with the boundaries of these 2 objects that we wanted to capture. And this is something that we cannot do easily with snakes. So, you can see, here we are not evolving the contour directly.

Actually, we are evolving this  $\phi$  function. So, from this binary, this binary step function to this function. This is evolved in such a way that the contours come close to the edges. This.

So, there are various versions of level set methods available online. You can check, check them. You can play with them. But our objective was to show the basic idea.

(Refer Slide Time: 24:08)

MATLAB Tutorial for Chan-Vese Segmentation

Refer to ~0:10 of lecture video

Mumford-Shah functional

$$F^{MS}(u, c) = F_A + F_B + F_C$$

$$F^{MS}(u, c) = \underbrace{\mu \text{length}(c)}_{\text{Keep } C \text{ smooth and compact}} + \underbrace{\lambda \int_{\Omega} |f - u|^2 dx dy}_{\text{Like MSE Explained in ~3:00 of lecture video}} + \underbrace{\int_{\Omega/C} |\nabla u|^2 dx dy}_{\text{Avoid too many edges in the region bounded by } C}$$

where

$f$ : Image,  $u$ : Piecewise smooth approximation of  $f$

NPTEL

MATLAB Tutorial for Chan-Vese Segmentation

Refer to ~0:10 of lecture video

Mumford-Shah functional

$$F^{MS}(u, c) = F_A + F_B + F_C$$

$$F^{MS}(u, c) = \underbrace{\mu \text{length}(c)}_{\text{Keep } C \text{ smooth and compact}} + \underbrace{\lambda \int_{\Omega} |f - u|^2 dx dy}_{\text{Like MSE Explained in ~3:00 of lecture video}} + \underbrace{\int_{\Omega/C} |\nabla u|^2 dx dy}_{\text{Avoid too many edges in the region bounded by } C}$$

where

$f$ : Image,  $u$ : Piecewise smooth approximation of  $f$

$C$ : Evolving curve in  $\Omega$

$\Omega$ : Comp domain,  $\Omega/C$ : domain inside " $C$ "

NPTEL

So, now, I come to the last part, the last model. It is called Chan-Vese model. So, this is also a level set based active contour model but the difference is in the value of functional. So, here instead of using an edge attractor, the functional used is a, it is called a Mumford-Shah function and it contains 3 terms  $F_A, F_B, F_C$ . So, here, if  $F$  is a image, small  $f$  and  $u$  is its piecewise smooth approximation and  $C$  is an evolving curve in  $\gamma$ , some computational domain.

Then, and also omega by C, omega over C represents domain inside C. So, with this notation, the formula of a Mumford-Shah functional looks like this. So, what it does is the first term  $F_A$ , it penalizes, so, if you have a long length, if the length of the contour is long, it you will penalize. So, its objective is to keep C smooth and compact. The third term  $F_C$ , it penalizes for edges within segmented region.

So, for example, you have segmented some region with the help of a contour C. So, within that region, if you have a lot of gradients, if you have a lot of gradients that means there are some edges. So, it penalizes if you have too many edges in the region. So, the objective is to avoid too many edges in the region bounded by the C.

Now, the second term  $F_B$ , it looks like mean square error. So, it penalizes, if the piece by the approximation is very different from the original image. So, this is the main feature of this function like lambda times f minus u square, the  $F_B$  term. So, sir has explained this in 3 minutes in the lecture video. So, I will just take the same example.

(Refer Slide Time: 26:51)

Refer to ~ 3:00 of lecture video

$F_1(C) > 0, F_2(C) \approx 0$   
Fitting > 0

$F_1(C) \approx 0, F_2(C) > 0$   
Fitting > 0

$F_1(C) > 0, F_2(C) > 0$   
Fitting > 0

$F_1(C) \approx 0, F_2(C) \approx 0$   
Fitting = 0

**Fig. 1.** Consider all possible cases in the position of the curve. The fitting term is minimized only in the case when the curve is on the boundary of the object.

$F_B = F_1(c) + F_2(c)$

$= \int_{\Omega_1} |f - u_1|^2 dx dy + \int_{\Omega_2} |f - u_2|^2 dx dy$

$\swarrow$  Constants  $\int_{\Omega_i} u_i dx dy$   
 $u_1 = \frac{\int_{\Omega_1} f dx dy}{\text{Ar}(\Omega_1)}$   
 $u_2 = \frac{\int_{\Omega_2} f dx dy}{\text{Ar}(\Omega_2)}$

Mumford-Shah functional

$$F^{MS}(u, C) = F_A + F_B + F_C$$


$$F^{MS}(u, C) = \underbrace{\mu \text{length}(C)}_{\substack{\text{Keep } C \text{ smooth} \\ \text{and compact}}} + \underbrace{\lambda \int_{\Omega} |f - u|^2 dx dy}_{\substack{\text{Like MSE} \\ \text{Explained in } \sim 3:00 \\ \text{of lecture video}}} + \underbrace{\int_{\Omega/C} |\nabla u|^2 dx dy}_{\substack{\text{Avoid too many edges} \\ \text{in the region bounded} \\ \text{by } C}}$$

where

$f$ : Image,  $u$ : Piecewise smooth approximation of  $f$

$C$ : Evolving curve in  $\Omega$

$\Omega$ : Comp domain,  $\Omega/C$ : domain inside "C"



So, here you can see, here you just look at this figure, this expression, here  $f$  minus  $u$ . So, here  $u$ , what we have taken for  $u$  is, here,  $u$  is simply the average of image intensity in the segmented region. So, for example, here we have  $C$ , this white curve. So, within that we have  $\Omega_1$ , outside it is  $\Omega_2$ . So, whatever is average within  $\Omega_1$  that is  $\mu_1$ .

And then we are calculating  $f - \mu_1$ . So, considering all possible options, you can see that the value of this energy function will drop to 0 only when the curve fits the actual boundary. So, in this case you can see, the first  $F_1, F_1$  will be greater than 0,  $F_2$  will be close to 0. Because in  $F_2$ , everything is smooth. In this case,  $F_1$ , because most of the area is black, so,  $F_1$  is almost zero. But outside we have black and this gray. So,  $F_2$  will be greater than zero.

Here in this case you see, there is black and white in both region,  $\Omega_1$  also,  $\Omega_2$  also. So, that is why it is also greater than 0. So, it will be fitting well only when we have this type of structure and this is what we want to do. So, if you consider all possible cases in the position of curve, the fitting term is minimized only in the case when the curve is on the boundary of the object. So, this explains this functional.

So, just like in the last 2 approaches, once we have a functional but this functional is written in terms of  $C$  and we want to use level set formulation. So, we need to find a way to write this functional in the form of  $\phi$ .




(Refer Slide Time: 29:07)


Refer to ~ 5:24 of lecture video

Level Set Formulation

Given constraints on "C", the energy functional in terms of  $\phi$  is as follows:

$$F(\mu_1, \mu_2, \phi) =$$
$$\mu \int_{\Omega} S(\phi(x,y)) |\nabla \phi(x,y)| dx dy +$$

$$\mu \int_{\Omega} S(\phi(x,y)) |\nabla \phi(x,y)| dx dy +$$
$$\lambda_1 \int_{\Omega} |f(x,y) - \mu_1|^2 H(\phi(x,y)) dx dy +$$
$$\lambda_2 \int_{\Omega} |f(x,y) - \mu_2|^2 (1 - H(\phi(x,y))) dx dy$$

H is heaviside function. Because it is not differentiable, its smooth version should be used. (Refer ~ 6:00)



So, in the level set formulation, given constraints on C, the energy functional in terms of  $\phi$  can be written as follows. Here, H, H is heaviside function. So, because it is not differentiable, its smooth version should be used. So, please, refer to this timestamp in the lecture to know what smooth version should we use.

(Refer Slide Time: 29:36)


Refer to ~ 5:24 of lecture video

Level Set Formulation

Given constraints on "C", the energy functional in terms of  $\phi$  is as follows:

$$F(\mu_1, \mu_2, \phi) =$$

$$\mu \int_{\Omega} \delta(\phi(x,y)) |\nabla \phi(x,y)| dx dy +$$


$$\lambda_1 \int |f(x,y) - \mu_1|^2 H(\phi(x,y)) dx dy +$$


Refer to ~ 6:22 of lecture video

The corresponding Euler-Lagrange equation

$$\frac{\partial \phi}{\partial t} = \delta(\phi) \left[ \mu \nabla \cdot \left( \frac{\nabla \phi}{|\nabla \phi|} \right) - \lambda_1 (f - \mu_1)^2 + \lambda_2 (f - \mu_2)^2 \right]$$

$\downarrow$   
 $\frac{1}{\pi} \frac{\epsilon}{\epsilon^2 + \phi^2}$   
 (Smooth Dirac delta function)



13 | of 16

MIA\_ActiveContour\_Tutorial.pdf

130.6%

Refer to ~ 3:00 of lecture video

$F_1(C) > 0, F_2(C) \approx 0$   
Fitting > 0

$F_1(C) \approx 0, F_2(C) > 0$   
Fitting > 0

$F_1(C) > 0, F_2(C) > 0$   
Fitting > 0

$F_1(C) \approx 0, F_2(C) \approx 0$   
Fitting = 0

Fig. 1. Consider all possible cases in the position of the curve. The fitting term is minimized only in the case when the curve is on the boundary of the object.

$F_B = F_1(c) + F_2(c)$

$= \int_{\Omega_1} |f - \mu_1|^2 dx dy + \int_{\Omega_2} |f - \mu_2|^2 dx dy$

$\mu_1 = \frac{\int_{\Omega_1} f dx dy}{\text{Ar}(\Omega_1)}$   
 $\mu_2 = \frac{\int_{\Omega_2} f dx dy}{\text{Ar}(\Omega_2)}$

Constants

NPTEL

So, once we have this functional in terms of  $\phi$ , we can again use the same Euler-Lagrange equation and find the, this gradient decent equation. So, this is what we get.  $\partial\phi/\partial t$  is equal to this Dirac delta times this function. So, this is again, because we want to use smooth functions, so, this Dirac delta function is smooth with the help of this function.

$$\frac{1}{\pi} \frac{\epsilon}{\epsilon^2 + \phi^2}$$

So, now, I will take, now, because this formulation is shown to you, in the MATLAB snippet, we will show you the mean square loss term. This term that we were explaining for so long. How this is implemented in the Chan-Vese segmentation.

(Refer Slide Time: 30:44)

delta function)

### MATLAB Snippets

% Coded by: Shawn Lankton (www.shawnlankton.com)

```

%-- 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(idxx)-u).^2-(I(idxx)-v).^2; % force from image information
curvature = get_curvature(phi,idxx); % force from curvature penalty
dphidt = F./max(abs(F)) + alpha*curvature; % gradient descent to minimize energy

```

~ 3:00

NPTEL

Fitting > 0

Fig. 1. Consider all possible cases in the position of the curve. The fitting term is minimized only in the case when the curve is on the boundary of the object.

Refer to ~ 5:24 of lecture video

NPTEL

### Mumford-Shah functional

$$F^{MS}(u, c) = F_A + F_B + F_C$$

$$F^{MS}(u, c) = \underbrace{\mu \text{length}(c)}_{\text{Keep } c \text{ smooth and compact}} + \underbrace{\lambda \int_{\Omega} |f - u|^2 dx dy}_{\text{Like MSE Explained in ~3:00 of lecture video}} + \underbrace{\int_{\Omega/c} |\nabla u|^2 dx dy}_{\text{Avoid too many edges in the region bounded by } c}$$

where

$f$ : Image,  $u$ : Piecewise smooth approximation of  $f$

$C$ : Evolving curve in  $\Omega$

$\Omega$ : Comp domain,  $\Omega/c$ : domain inside "C"

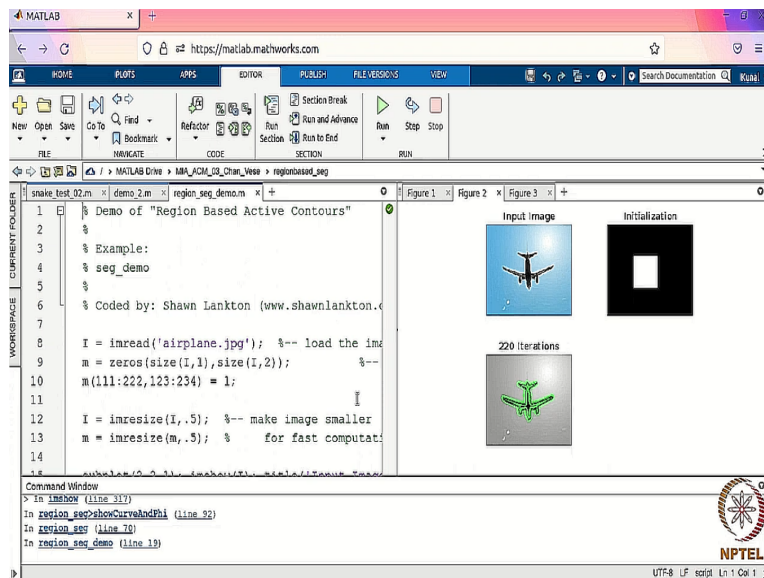
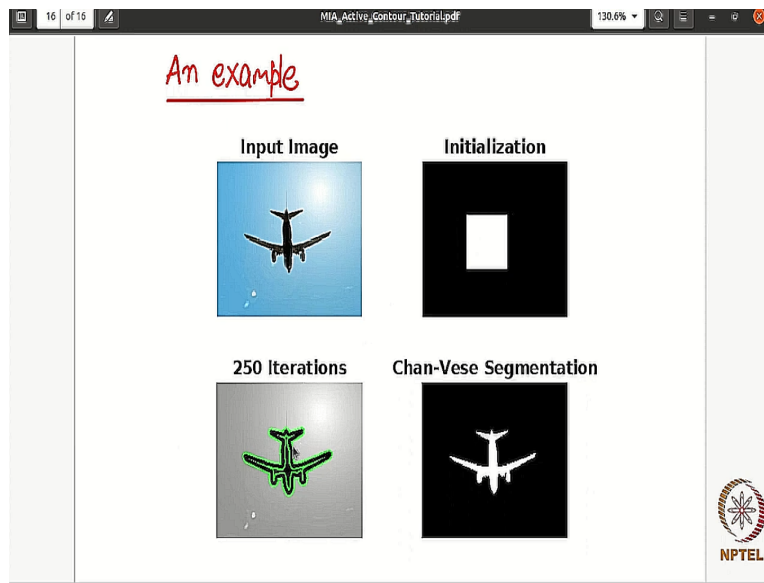
NPTEL

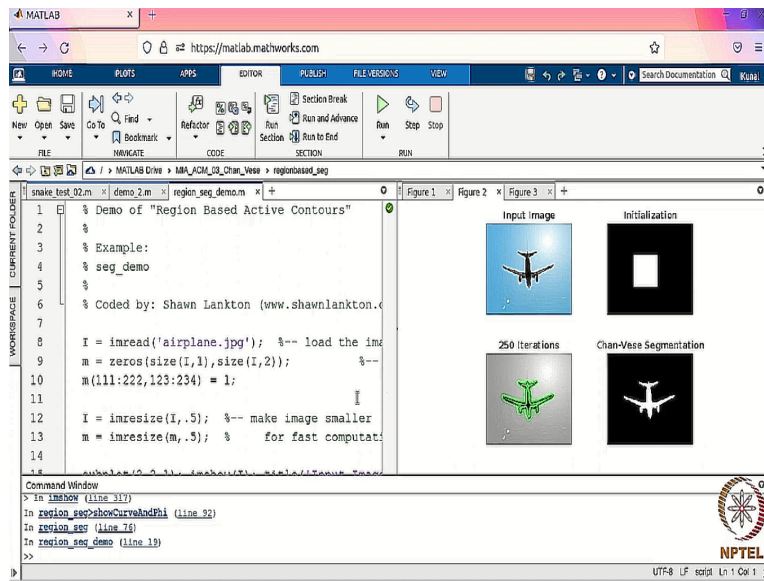
So, this code is by Shawn Lankton and it is available in MATLAB. It is available in MATLAB central. So, in this snippet you can see like with the help of level set we can first find the interior points then exterior points and then we are calculating the mean of that, mean of these points,  $u$  and  $v$ . So, this is basically  $\mu_1$  and  $\mu_2$  in our expression here, the thing this  $\mu_1, \mu_2$ .

So, we are calculating in MATLAB like this. And then this formula, the force from image information, you see,  $I$  minus  $u$  square plus  $I$  minus  $v$  square. So, this is same as this,  $f$  minus  $u$ . So, you can see, it also like, it has some other terms also get curvature etcetera. So, that I do not know.

So, but it has some more terms also. But like in the previous examples, we can see that whatever we have learned in this course is actually sitting in the advanced code also. And you can like play with parameters, you can modify the code like simplify the code for your purpose also or you can even make like add your own terms and see if the results improve or not. So, you can play with this code. But the main thing is whatever we are learning here is actually deployed in these models.

(Refer Slide Time: 32:17)





So, here is an example, an example of a Chan-Vese implementation segmentation. So, you have an initial image of this aircraft. You initialize it with this. So, this is your initial level set and then after some iterations you get this. So, I will show you. So, if you run this, you see, this is an input image. Yes, this is your initial level set and after some time, you can see, it is able to like capture the boundaries of this aircraft, yeah. So, thanks everyone.