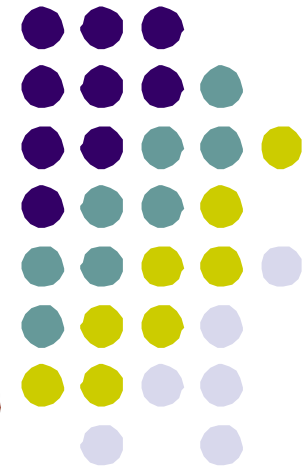
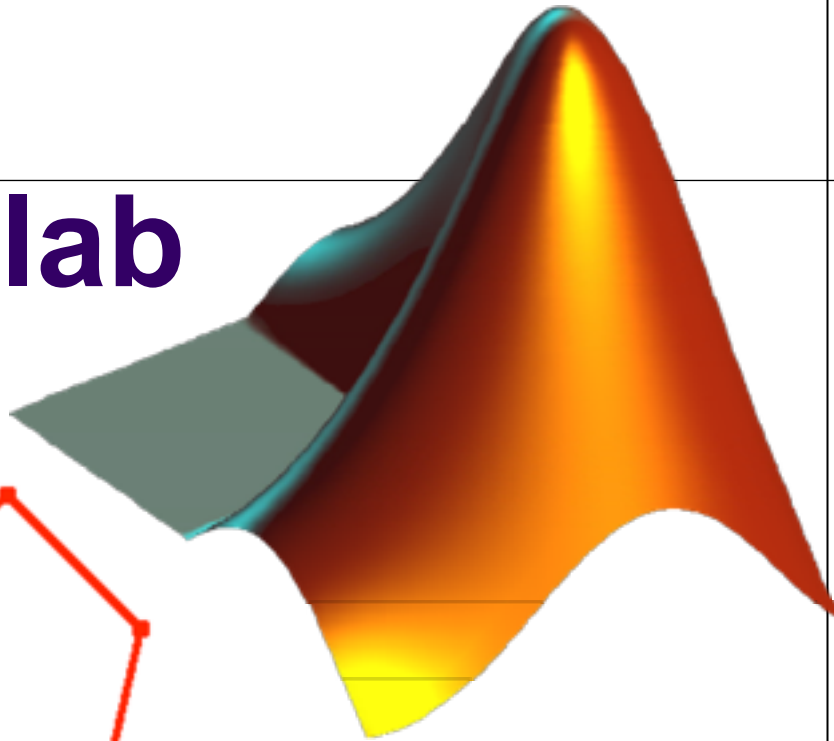
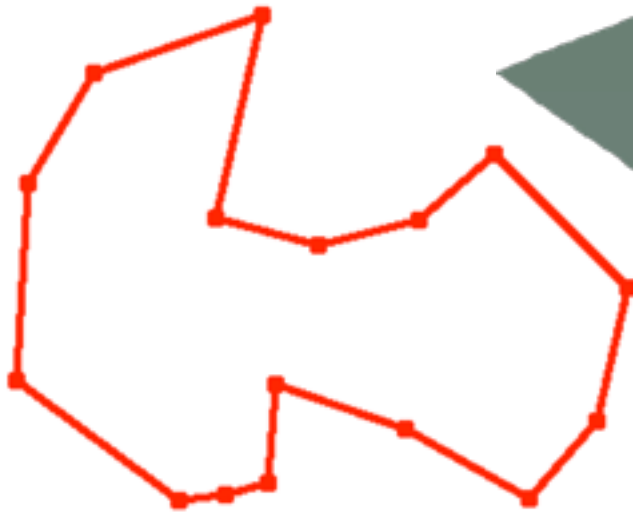
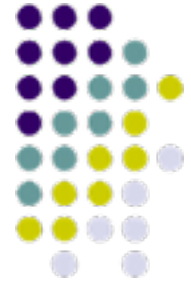


Snakes with Matlab



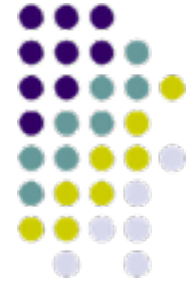
Outline



- Today we will experiment with snakes (active contours) technique to perform image segmentation



Outline



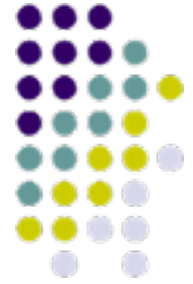
- Today we will experiment with snakes (active contours) technique to perform image segmentation

Specifically, we will see:

- How to interactively update a plot in Matlab
- How to pick a set of point from an image
- Subplots (useful to simultaneously display the snake energy evolution and its shape)

And some math:

- Snake energy functional
- The energy gradient
- Discuss some implementation details



Demo time

Image loading and displaying



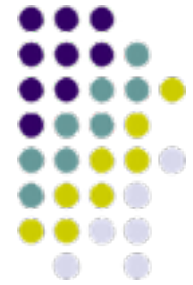
- We have already seen how to load an image into a Matlab matrix:

```
I = imread('image.png');
```

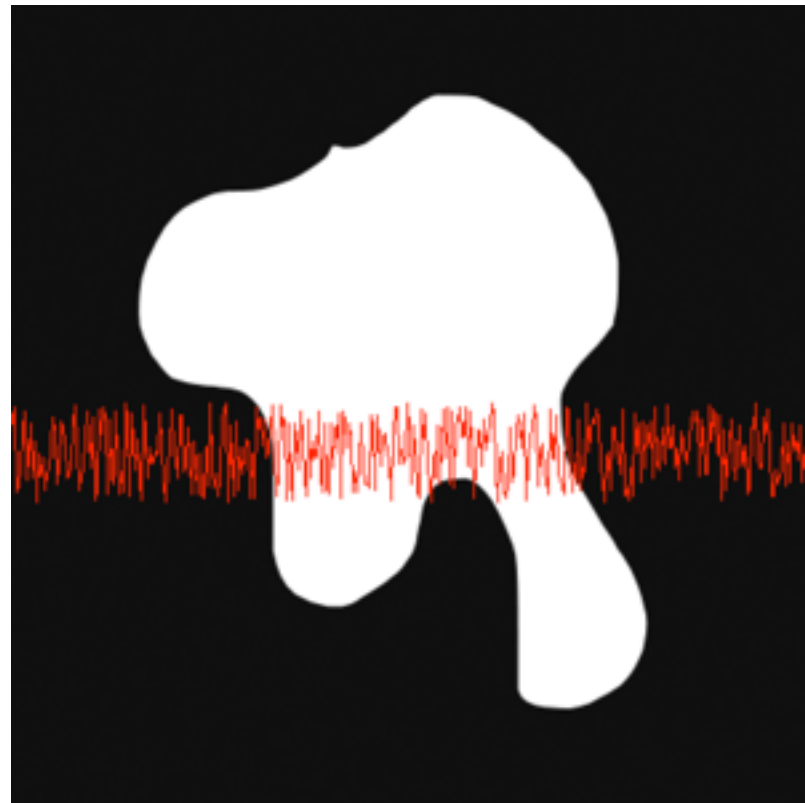
- The image I can then be easily displayed with the function **imshow**:

```
imshow(I);
```

Image loading and displaying



- Multiple plots can be stacked together by using the command **hold on**:



```
imshow(Im) ;  
hold on  
plot( rand(size(Im,2),1)*50+size(Im,1)/2, 'r' )
```

Plot update



- In many situations, it is useful to quickly change plot content without creating a new figure
 - For instance, to animate the evolution of a data serie
- To do this, we change the internal properties of a plot object via the **set** function and then invoke the **drawnow** command
- We need:
 - The identifier (in Matlab is called handle) of the object we wish to modify
 - The name of the property to update

Plot update example



```
x = linspace(0,2*pi,20);
```

plot function returns the handle of the graphics object

```
h = plot(x,sin(x) );
```

```
for ph=0:100
```

by using the **set** function, we can change the YData and XData property of the plot

```
y = sin(x+0.1*ph*pi);
```

```
set(h, 'YData', y);
```

```
set(h, 'XData', x);
```

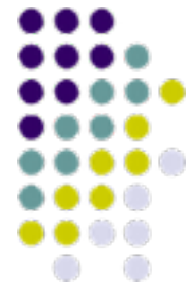
```
drawnow;
```

```
end
```

drawnow function forces the figure redraw

Can you guess the result?

Picking points from figure



- Matlab offers a simple way to interactively gather 2D point coordinates from figures

- coordinates of the picked points
- button pressed 1=left, 2=middle, 3=right
- number of points to pick

[x y b] = ginput(n);

Picking points from figure Example



```
[x, y, but] = ginput(1);  
points = [x;y];  
h = scatter( points(1,:), points(2,:) );
```

```
while but == 1  
    [x, y, but] = ginput(1);  
    points = [ points [x;y] ];  
    set(h, 'YData', points(2,:));  
    set(h, 'XData', points(1,:));  
    drawnow;  
end
```

- May be useful to force the axis limits before calling ginput to avoid figure rescaling while selecting points

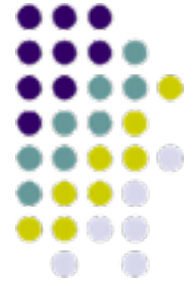
Subplots



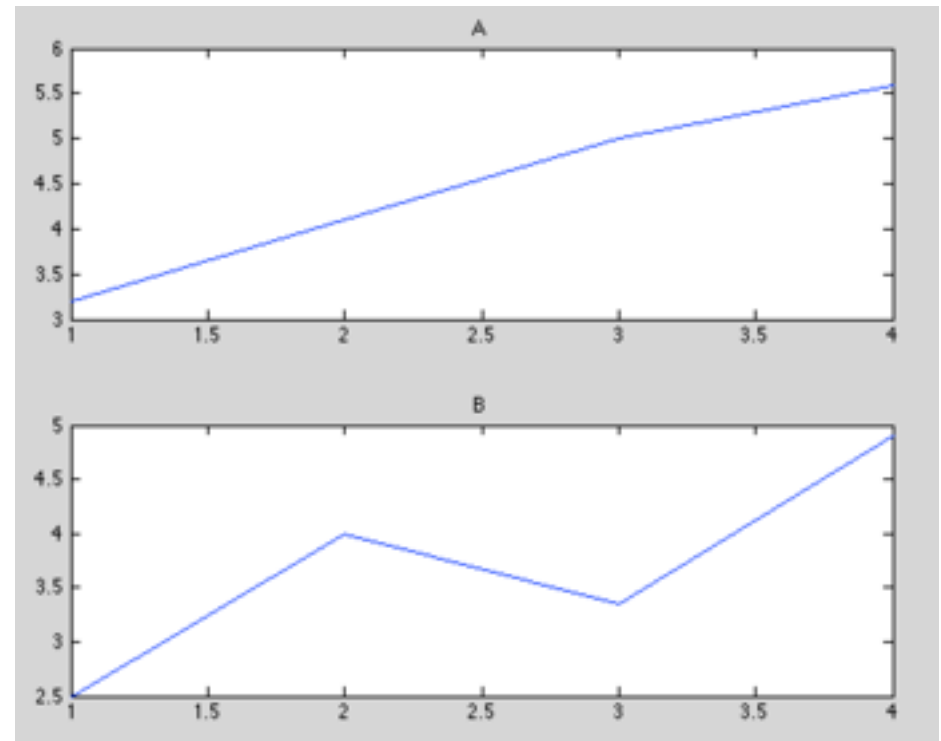
- subplot command can be used to subdivide the current figure into a grid. Each grid element can contain a different type of plot
 - Grid height (number of rows)
 - Grid width (number of columns)
 - p^{th} position to use for the new plot

`subplot (m, n, p) ;`

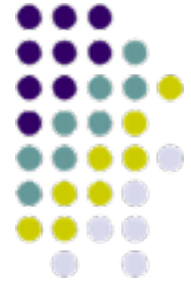
Subplots Example



```
A = [3.2, 4.1, 5.0, 5.6];  
B = [2.5, 4.0, 3.35, 4.9];  
subplot(2,1,1); plot(A)  
title('A')  
subplot(2,1,2); plot(B)  
title('B')
```



Snakes

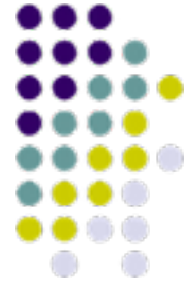


- Snakes are energy minimising curves that deform to fit image features
 - In our example, we are interested to let such curve being attracted to image edges
- We parametrize a 2D curve as a function

$$v(s) = \begin{bmatrix} x(s) \\ y(s) \end{bmatrix}, s = 0 \dots 1$$

- Its shape is influenced by internal forces (i.e. curvature, length etc) and image forces that attracts the curve toward interesting features

Snake energy functional



- We let the snake curve evolve to minimize the following energy functional:

$$E_{snake}(v(s)) = \alpha \int_0^1 E_{cont}(v(s))ds + \beta \int_0^1 E_{curv}(v(s))ds + \gamma \int_0^1 E_{img}(v(s))ds$$

$\left| \frac{dv}{ds} \right|^2$ $\left| \frac{d^2v}{ds^2} \right|^2$ $- |\nabla I(v(s))|^2$

Three red arrows point from the terms below to the integrands in the equation above: from $\left| \frac{dv}{ds} \right|^2$ to E_{cont} , from $\left| \frac{d^2v}{ds^2} \right|^2$ to E_{curv} , and from $- |\nabla I(v(s))|^2$ to E_{img} .

Energy gradient

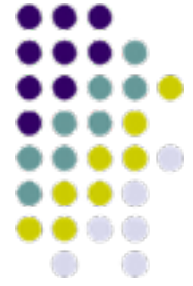


- The initial shape of the curve is defined by the user close to the edges of interest
- Snake energy can be minimised by gradually altering each variable by a small quantity $d\mathbf{v}$ **in the direction of the negated gradient** of E_{snake}
- For small variations, we can write:

$$E_{snake}(\mathbf{v}) = \alpha \int_0^1 |\mathbf{v}'|^2 ds + \beta \int_0^1 |\mathbf{v}''|^2 ds + \gamma \int_0^1 E_{img}(\mathbf{v}) ds$$

$$E_{snake}(\mathbf{v} + \delta\mathbf{v}) = \alpha \int_0^1 |\mathbf{v}' + \delta\mathbf{v}'|^2 ds + \beta \int_0^1 |\mathbf{v}'' + \delta\mathbf{v}''|^2 ds + \gamma \int_0^1 E_{img}(\mathbf{v} + \delta\mathbf{v}) ds$$

Energy gradient



- With the following approximations:

$$E_{img}(\mathbf{v} + \delta\mathbf{v}) = E_{img}(\mathbf{v}) + \frac{\delta E_{img}}{\delta\mathbf{v}} \delta\mathbf{v}$$

$$|\mathbf{v} + \delta\mathbf{v}|^2 = \mathbf{v}\mathbf{v} + 2\mathbf{v}\delta\mathbf{v} + \delta\mathbf{v}\delta\mathbf{v} \approx \mathbf{v}^2 + 2\mathbf{v}\delta\mathbf{v}$$

... and after some manipulations and integration by parts, we obtain the energy gradient:

$$\frac{\delta E_{snake}(\mathbf{v})}{\delta\mathbf{v}} = \frac{E_{img}}{\delta\mathbf{v}} - \alpha\mathbf{v}'' + \beta\mathbf{v}''''$$

Energy gradient



- Where, if we consider:

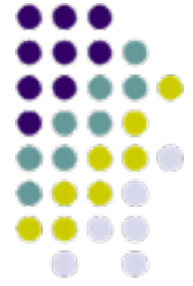
$$E_{img} = - |\nabla I(v(s))|^2$$

We have:

$$-\frac{E_{img}}{\delta \mathbf{v}} = \frac{\delta}{\delta \mathbf{v}} (|\nabla I(\mathbf{v})|^2) = 2 \underline{\nabla \nabla I(\mathbf{v})} \nabla I(\mathbf{v})$$

Hessian matrix of second order image derivatives

Our snake implementation



- We implement our snake as:
 - A discrete set of n 2D points

$$S = [\mathbf{x}_1 \dots \mathbf{x}_n] = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \end{bmatrix}$$

- Closed curve (i.e. the beginning and the end of the curve are on the same point)
- In our discrete case, the energy functional reduces to a summation over each snake point

Our snake implementation



- Energy functional:

$$E_{snake}(S) = \alpha \sum_{i=1}^n E_{cont}(\mathbf{x}_i) + \beta \sum_{i=1}^n E_{curv}(\mathbf{x}_i) + \gamma \sum_{i=1}^n E_{img}(\mathbf{x}_i)$$

- The initial shape of the curve is defined by the user by manually picking a set of points close to the edges of interest
- Snake energy can be minimised by iteratively:
 - Computing the energy gradient dE_i at each point $\mathbf{x}_i$
 - Subtracting to each $\mathbf{x}_i$ the value of dE_i multiplied by a small constant *step_size*

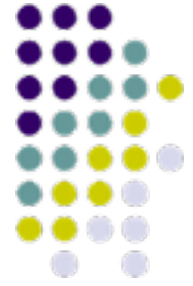
Our snake implementation



- We need to:
 - Compute the 1st, 2nd and 4th order partial derivatives of the snake curve
- **Solution:** Use a discrete approximation using finite differences. For example, to compute the n^{th} order central finite difference with step h :

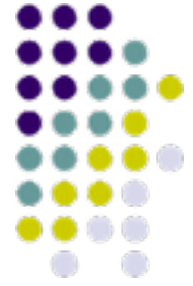
$$\delta_h^n[f](x) = \sum_{i=0}^n (-1)^i \binom{n}{i} f\left(x + \left(\frac{n}{2} - i\right)h\right)$$

Our snake implementation



- We need to:
 - Compute the image gradient and hessian matrix for each pixel
- **Solution:** Convolve with a kernel like derivative of gaussian

Our snake implementation



- We need to:
 - Sample an image (matrix) at certain non-integer locations
- **Solution:** Use matlab interp2 function. Type help interp2 to get a precise description of how to use it