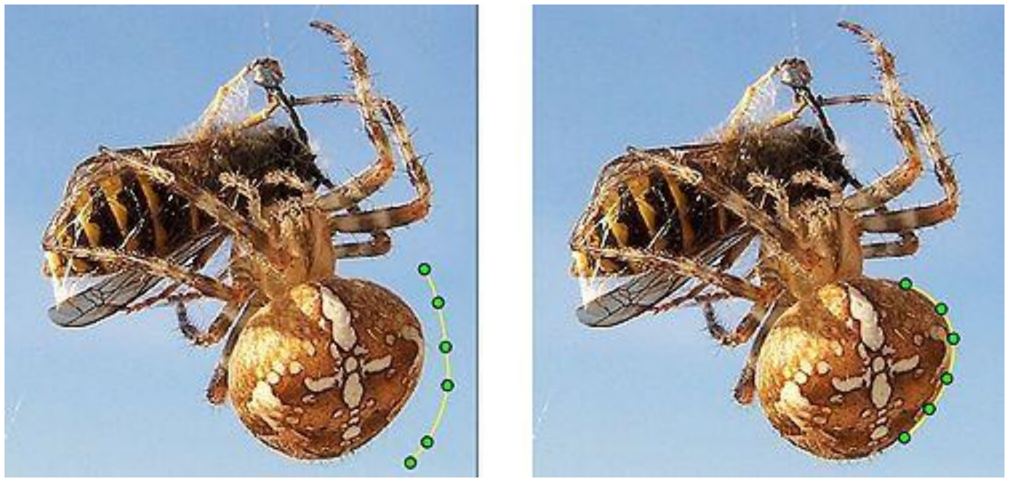
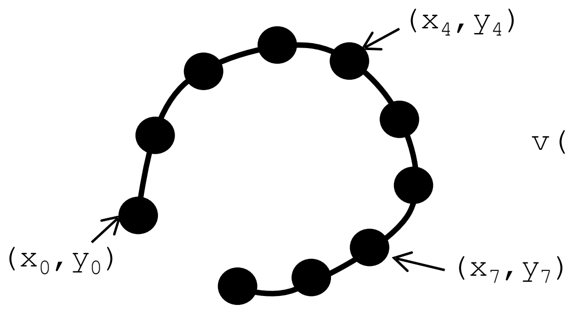
Lecture notes: Active contours

An active contour, also known as a "snake", is a semi-automated method for segmentation. It takes an initial contour and iteratively moves it based upon a set of criteria. The goal is to start with an initial contour close to a desired location, and then use the active contour algorithm to drive the contour to the final desired location.



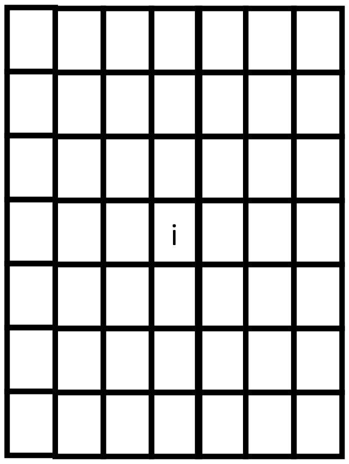
The image above (take from the Wikipedia article on active contours) shows an active contour at two points in time. In the left image, the initial contour is shown placed to the bottom right of the spider. In the right image, the contour is shown after applying the active contour algorithm. Notice that the contour has moved to the boundary of the spider, which is the desired segmentation location.

The active contour algorithm works as follows. The contour v is modeled with a discrete set of points:



$v(i)=(x_i,y_i)$ for $i=1\ldots n$

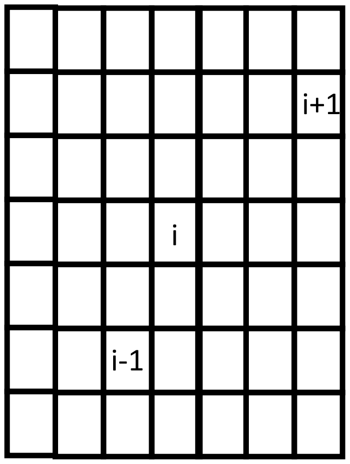
The contour is moved by searching in the neighborhood of each pixel on the contour for a better location. For example, suppose we are searching for a new location for pixel i in a 7x7 window:



calculate energy at
each pixel in window;
contour point i moves
to pixel with
lowest energy

For every location in the window, we calculate **internal** and **external** energy terms. The contour point i is then moved to the location that has the smallest total energy.

Internal energy terms are defined by constraints upon the shape of the contour. They are not dependent on the image data, they only depend upon the locations of the other contour points. For example, suppose the previous and next contour points look like this:



We can define the internal energy as the square of the distance between points:

$$E_{internal} = (x_i - x_{i+1})^2 + (y_i - y_{i+1})^2$$

At its current location, the contour point i has an internal energy equal to $3^2+2^2=13$. Across the entire window, the internal energy at point i looks like the following:

37	26	17	10	5	2	1
36	25	16	9	4	1	0
37	26	17	10	5	2	1
40	29	20	13	8	5	4
45	34	25	18	13	10	9
52	41	32	25	20	17	16
61	50	41	34	29	26	25

The internal energy for point i is at its minimum if point i is relocated to point i+1, hence the value of 0 at that location. As the point i is moved farther from i+1, the value of the internal energy gets larger.

External energy terms depend only upon the image data, and are not affected by the shape of the contour. For example, assume the window has the following greyscale image data:

11	24	27	23	11	8	5
10	22	25	25	10	9	5
12	20	27	20	12	8	4
10	20	24	22	10	8	5
9	21	29	20	11	7	5
10	24	23	25	10	6	6
11	20	20	27	14	8	5

We can define the external energy as the negative of the brightness:

$$E_{external} = -I(x_i, y_i)$$

where I() is the image intensity. The negative is taken so that minimizing it moves it towards the brightest location.

The total energy at each pixel is the sum of the internal and external energies:

26	2	-10	-13	-6	-6	-4
26	3	-9	-16	-6	-8	-5
25	6	-10	-10	-7	-6	-3
30	9	-4	-9	-2	-3	-1
36	13	-4	-2	2	3	4
42	17	9	0	10	11	10
50	30	21	7	15	18	20

The internal and external energies each pull the point in a given direction. The internal energy pulled the point i upwards and to the right, while the external energy pulled the point leftwards and down. The sum of the pulls yields the new contour point location. For this example, the best location has a total energy of -16 and is up 2 pixels from its current location.

This process gets repeated for every point i in the contour. An iteration of the active contour completes once every point has moved. Typically, the active contour algorithm runs for 10-100 iterations. It can also be stopped once the total or average distance moved by all the contour points falls below a threshold.

Formally, the energy of an active contour may be written as:

$$E\big(v(i)\big) = \int_i E_{internal} + E_{external}$$

where V(i) is the contour. The goal is to find the set of image coordinates i that minimize the energy E(v(i)).

In the above example, we defined the energy terms so that the contour acted like a rubber band stretching towards the brightest parts of an image. The internal energy was defined to pull all the contour points towards each other, collapsing at a single point. The external energy was defined to pull the contour towards bright pixels.

We can however define the energy terms according to any desired criteria. Some common internal energy terms include:

- minimizing the average distance between points (rewarding even spacing of points)
- minimizing the curvature of points (rewarding points that make a straighter line)
- minimizing the difference of each point from an average curvature (rewarding points that make an arc of equal curvature)

For the above example, if we are trying to evenly space the points, the minimum energy would occur at the location to the right of i in the window. An energy term trying to make the points linear would also be minimum at that location. At all other locations (including the current location of i) the energy terms would have larger values.

Some common external energy terms include:

- minimizing the negative of the gradient (rewarding points that move towards edges)
- minimizing the variance of color of points (rewarding points that have the same color)
- minimizing variance of color within a closed contour (rewarding points that homogenize the enclosed contour)
- minimizing deviation in edge strength (rewarding points that have a similar gradient, either locally or globally)

An active contour can also have **multiple** internal and external energy terms, with each pulling the contour in a different manner.

Energy terms are usually **squared**, to provide a steeper gradient towards minima.

The **window size** must be large enough so that the desired image content can have an effect through external energy terms, but not so large that it allows the contour to get pulled to an alternate local minimum. Usually the window size is balanced against the **number of iterations**.

All the energy terms are usually **normalized** within the window, so that they carry equal distributions. This can be done simply by calculating the min and max values for an energy term within the window and normalizing across that range, or by specifying an image-wide min and max for each energy term so that the normalizations are consistent point to point.

The energy terms may be **weighted** differently, depending on the desired behavior of the active contour. Sometimes all the internal energy terms are grouped together with one weight versus all the external energy terms with a second weight, providing one simpler control.

The **initial** contour is typically supplied by a person manually drawing it. This is why the method may be considered semi-automated segmentation. It can however also be supplied by an automated method that creates a rough segmentation or boundary, where the purpose of the active contour algorithm is to further refine the result.

Active contours are **most useful** for problems where part of the desired boundary is well delineated, and part of the boundary is not well delineated. For example, consider the spider. The bottom of its body is easily found by color or gradient, but the top and legs of its body are less obvious. An initial contour could be drawn roughly around or near the entire body, and then the active contour algorithm could be applied to iterate the contour to a better fit.

In class, several demonstrations of active contours will be shown.