TP1: Active Shape Models

Loic Simon Chaohui Wang Panagiotis Koutsourakis

7 Mai 2007

1 Programming using the *CImg* library

The `CImg` library is an open source C++ toolkit for image processing (`http://cimg.sourceforge.net/index.shtml`). Due to its usefulness and simplicity, we use it throughout our lab exercises. The goal of this exercise is to learn how to use the `CImg` library, especially the `CImg` class, by realizing several tasks.

You first download all the necessary files from `http://www.mas.ecp.fr/vision/Personnel/juan/teaching/computer-vision/`.

1.1 "Hello, Lena"

In this part, based on the `CImg` library and the test image "lena.png", we will accomplish the sub-tasks as follows:

1. Construct an `CImg<double>` object which reads the image from the file "lena.png" and display the image of Lena. Try to use the methods of both the `CImg` class and the `CImgDisplay` class to display Lena.
2. Get and show the size (height and width) of the image using the method of `CImg<double>` class.
3. Display the `R` (Red) component of the image (hint: `channel` or `get_channel`).
4. Compute the gradient of the `R` component of the image along the x direction and display the result. Try to compute the gradient along the y direction if you have time.

1.2 National flag of France

Now,we will create an image representing the national flag of France.

1. Construct an `CImg<unsigned char>` object which contains a color image of size 600×400.
2. Fill the three channels (R, G, B) of the image by the colors representing the flag of France (blue(0,0,255), white(255,255,255) and red(255,0,0)).
3. Display the image and save it with format png, jpg or bmp in the hard disk.

2 Snake

In this section, we first review the theory on `snake`. And then we should do some exercises concerning the implementation of this algorithm.

2.1 continuous formulation

Here, we present the Kass' model, which is a common model of `snake`. In this model, a snake v is a parametric curve :

$$\begin{array}{rccl} v: & [0,1] & \rightarrow & \mathbb{R}^2 \\ & s & \rightarrow & v(s) = (x(s), y(s)) \end{array} \tag{1}$$

Lets define $\dot{x} \equiv \frac{dx(s)}{ds}$ $\ddot{x} \equiv \frac{d^2x(s)}{ds^2}$ and the same in y direction. Next, we express a smooth closure condition on the contour:

$$x(0) = x(1), \dot{x}(0) = \dot{x}(1), \ddot{x}(0) = \ddot{x}(1)$$

.

Finally, we define the energy of the snake as a two terms sum:

$$E_{total}(v) = E_{int}(v) + E_{ext}(v) \tag{2}$$

The first term is defined as follows, which ensures that a snake with a low energy is regular (low curvature):

$$E_{int}(v) = \int_0^1 w_{elast} \frac{1}{2}(\dot{x}(s)^2 + \dot{y}(s)^2) + w_{stiff} \frac{1}{2}(\ddot{x}(s)^2 + \ddot{y}(s)^2) ds \tag{3}$$

The external term penalizes snakes placed in uniform areas of the input image, and is defined as:

$$E_{ext}(v) = w_{ext} \int_0^1 P(x(s), y(s)) ds \tag{4}$$

In our case, P should have low value near the contour in the image, then can be as:

$$P(x,y) = \frac{1}{1 + a\|\nabla(G_\sigma * I)(x,y)\|^p} + \epsilon \tag{5}$$

where $*$ stands for convolution and G_σ is a centered Gaussian with σ as standard deviation.

We're looking for the snake which minimizes the global energy:

$$v^{opt} = \arg\min_v E_{total}(v) \tag{6}$$

2.2 Discrete formulation

To implement the snake algorithm we need a discrete formulation. A simple method is provided below.

First we discretize the curve representation by assuming that a curve is determined by N_p control points:

$$(x_i, y_i) = (x(hi), y(hi)) \quad \forall i \in \{0, N_p - 1\}$$

Then we take the simplest discretization of the parameterization $s_i = hi$ and use a finite differences scheme to evaluate the derivative, we get:

$$\begin{aligned} E_{int}(v) = h \sum_{i=0}^{Np} [& w_{elast} \frac{1}{2h^2} ((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2) \\ & + w_{stiff} \frac{1}{2h^4} ((x_{i-1} - 2x_i + x_{i+1})^2 + (y_{i-1} - 2y_i + y_{i+1})^2)] \end{aligned} \tag{7}$$

and,

$$E_{ext}(v) = w_{ext} h \sum_{i=0}^{Np} P(x_i, y_i) \tag{8}$$

2.3 Minimization

We can use an iterative scheme to optimize the energy: gradient descent.

$$v_{t+1} = v_t - \gamma \nabla_v E_{total}(v_t) \tag{9}$$

3 Exercices

3.1 External energy

Complete the code of the Snake::computeP() function in snake.cpp.

3.2 Optimization

Complete the Snake::evolve() function in snake.cpp.

3.3 Test

Test the algorithm against rectangle.png. Comment the result.

3.4 Internal versus External energy

Test the algorithm with different values for the respective weight of the internal and external energies (to do so play with the three global variables wElas, wStiff and wExt defined at the top of snake.cpp). In particular try it respectively without one of the two components of the energy (wElast=wStiff=0 then wExt=0). What are the problems in each case?

3.5 Bluring the input image

Evaluate the behavior of the algorithm without bluring the input image (put sigma to 0 in snake.cpp). For this question, use rectangle.png and rectangleNoisy.png. What can you notice (benefits and drawbacks of the blur)?

3.6 Bonus question

What are the principal drawbacks of the current implementation. What can you propose to improve it.

References

[1] Kass M, Witkin A, Terzopoulos D. Snakes: active contour models. In: First international conference on computer vision; 1987. p. 259–68.

[2] Li, T. and Zhang, Y. and Yao, D. and Hu, D. FFT snake: a robust and efficient method for the segmentation of arbitrarily shaped objects in image sequences : ICPR 2004 p.II: 116-119