ECE 6560 – Partial Differiential Equations for image processing and computer vision

Final Project Report

28 avril 2008

Cyril Roussillon

Professor:

Anthony Yezzi

Georgia Institute of Technlology

Computer Science

Spring 08

1 Problem statement

A fundamental problem in computer vision is recognition of objects. A lot of machine learning techniques needs first a reduction of the complexity of the scene. Contour of an object is one of its most significative characteristics. Some interesting classification techniques even need an exact closed contour of the object (such as Fourier descriptors, curvogram analysis, ...), contrary to other techniques (such as Hough transform, some neural networks, ...).

Thus our goal is, given an image representing an object and its approximate position, to find its contour.

2 Mathematical formulation

Active contours (*a.k.a.* snakes) are an interesting solution to find closed contours of objects. Since their introduction, many variants have been developped, in particular edge-based active contours, region-based active contours, and Gradient Vector Flow active contours (though they do not derive from an energy functional).

There are a lot of applications where the contour is characterized by a strong gradient. Edge-based active contours uses this fact, and this is the one I implemented.

2.1 Energy functional

So our goal is to maximize the gradient on the curve, or more precisely the average of the gradient on the curve. Moreover contours of artificial objects are smooth, so we want a smooth curve. One way to have a smooth curve is to minimize its length, what can be done by maximizing the integral of the gradient on the curve instead of the average.

Let the following notations :
- $c(p) = \begin{pmatrix} x(p) \\ y(p) \end{pmatrix}$ the curve parameterized by $p \in [0, 1]$.
- $I(x, y)$ the greyscale image

Now let's consider the following function, called the conformal factor :

$$\phi(c(p)) = \frac{1}{1 + \|\nabla I(x(p), y(p))\|^2}$$

Clearly if the gradient is small, then $\phi \approx 1$, and if the gradient is large then $\phi \approx 0$.

Thus we want to minimize the following energy :

$$E(C) = \int_C \phi \cdot \mathrm{d}s$$

This energy is geometric, because it does not depend on the parameterization of the curve C, only the arclength.

2.2 Resolution

Suppose the curve evolves with time, then in order to minimize E, we have to descent its gradient.

$$
\begin{aligned}
\frac{\mathrm{d}E}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\int_C \phi\cdot\mathrm{d}s \\
&= \frac{\mathrm{d}}{\mathrm{d}t}\int_C \phi\left(c(p,t)\right)\cdot\underbrace{\left\|c_p\right\|\cdot\mathrm{d}p}_{\mathrm{d}s} \\
&= \int_0^1 \left[c_t\cdot\nabla\phi\cdot\left\|c_p\right\| + \phi\cdot\left\|c_p\right\|_t\right]\cdot\mathrm{d}p \\
&\quad \text{By the way : } \left\|c_p\right\|_t = \frac{\mathrm{d}}{\mathrm{d}t}\sqrt{c_p\cdot c_p} = \frac{c_{pt}\cdot c_p + c_p\cdot c_{pt}}{2\sqrt{c_p\cdot c_p}} = \frac{c_{pt}\cdot c_p}{\left\|c_p\right\|} \\
&\quad \text{And by integration by parts :} \\
&\quad \int_0^1 \phi\cdot\frac{c_{pt}\cdot c_p}{\left\|c_p\right\|}\mathrm{d}p = \int_0^1 \left(\frac{\phi\cdot c_p}{\left\|c_p\right\|}\right)\cdot c_{pt}\cdot\mathrm{d}p = \underbrace{\left[\frac{\phi\cdot c_p\cdot c_t}{\left\|c_p\right\|}\right]_{p=0}^1}_{0\text{ because } c \text{ is 1-periodic}} - \int_0^1 \left(\frac{\phi\cdot c_p}{\left\|c_p\right\|}\right)_p\cdot c_t\cdot\mathrm{d}p \\
&= \int_0^1 \left[c_t\cdot\nabla\phi\cdot\left\|c_p\right\| - c_t\cdot\left(\frac{\phi\cdot c_p}{\left\|c_p\right\|}\right)_p\right]\mathrm{d}p \\
&= \int c_t\cdot\left[\nabla\phi\cdot\left\|c_p\right\| - \underbrace{\frac{\left\|c_p\right\|}{\mathrm{d}s}}_{\frac{\mathrm{d}}{\mathrm{d}p}}\left(\phi\cdot\underbrace{\frac{c_p}{\left\|c_p\right\|}}_{c_s}\right)\right]\cdot\underbrace{\frac{\mathrm{d}s}{\left\|c_p\right\|}}_{\mathrm{d}p} \qquad (\text{because } \mathrm{d}s = \left\|c_p\right\|\mathrm{d}p) \\
&= \int c_t\cdot\left(\nabla\phi - (\phi\cdot c_s)_s\right)\mathrm{d}s \\
&= \int c_t\cdot\left(\nabla\phi - \phi_s\cdot\underbrace{c_s}_{T} - \phi\cdot c_{ss}\right)\mathrm{d}s \\
&= \int c_t\cdot\left(\underbrace{\nabla\phi - (\nabla\phi\cdot T)\cdot T}_{(\nabla\phi\cdot N)\cdot N} - \phi\underbrace{c_{ss}}_{\kappa\cdot N}\right)\mathrm{d}s \\
&= \left\langle c_t, \left(\nabla\phi\cdot N - \phi\cdot\kappa\right)\cdot N\right\rangle
\end{aligned}
$$

So the steepest descent is for :

$$c_t = \left(\phi\cdot\kappa - \nabla\phi\cdot N\right)\cdot N$$

I actually added an erosion/dilation term α :

$$c_t = \left(\phi\cdot(\kappa+\alpha) - \nabla\phi\cdot N\right)\cdot N$$

This allows to initialize the snake quite far away from the actual contour, inside or outside the contour.

Interpretation

– $\phi\cdot\kappa$ is the rigidity constraint. If the curvature κ is too big, it will be reduced. This rigidity constrained is stronger when there is little gradient (ϕ large).
– α is the propagation term. If $\alpha > 0$ we have dilation (the curve naturally tends to grow), and if $\alpha < 0$ we have erosion (the curve naturally tends to shrink). The actual term is $\phi\cdot\alpha$ because we want to reduce this natural propagation when the gradient is large (ϕ small). Note that the rigidity term $\phi\cdot\kappa$ already tend to erode the curve.

– $\nabla\phi \cdot N$ is the term that will tend to make the curve converging where ϕ is the smaller (the gradient is the larger). Actually its main effect will be to stop the curve when it is on a local extrema sufficiently large (when the curve is on a maximum gradient line, if it tries to go away this term will make it come back).

3 Implementation

3.1 Numerical implementation

I used marker particles to implement the gradient descent. So the curve is discretized into N points $c_i(p_i, t)$, and we want to compute :

$$c_t = (\phi \cdot \kappa - \nabla\phi \cdot N) \cdot N$$

with a temporal timestep Δt.

I used :

– upwind temporal differences, because we want to compute the contour at the next timestep according to the current contour,

– central spatial differences for the curve position (particles position), because we want to find the normal to the curve so we have to be centered around the point,

– central spatial differences for the image gradient, in order to be more stable (broader and more centered local view).

The computations are done in the image in pixels, width $\Delta x = \Delta y = 1$. The time step Δt is chosen so that the evolution of the curve is stable and fast enough.

The variables in capital upright characters I introduce (DX, DY, DDX, DDY, NORM, ...) are the temporary scalar variables I used in the code in order to keep it readable, and I reproduce them here for the same reason.

On the left hand size :

$$c_t = \frac{\partial c}{\partial t} = \frac{c(p, t + \Delta t) - c(p, t)}{\Delta t} = \frac{c_i(t + \Delta t) - c_i(t)}{\Delta t}$$

So the evolution with time of the curve is :

$$c_i(t + \Delta t) = c_i(t) + (\phi \cdot \kappa - \nabla\phi \cdot N) \cdot N \cdot \Delta t$$

Let's discretize this equation :

$$\begin{aligned} c_p &= \frac{c(p + \Delta p, t) - c(p - \Delta p, t)}{2\Delta p} = \frac{c_{i+1}(t) - c_{i-1}(t)}{2} \\ &= \begin{pmatrix} \mathrm{DX} \\ \mathrm{DY} \end{pmatrix} \\ c_{pp} &= \frac{c(t, p + \Delta p) - 2c(t, p) + c(t, p - \Delta p)}{(\Delta p)^2} = (c_{i+1}(t) - 2c_i(t) + c_{i-1}(t)) \\ &= \begin{pmatrix} \mathrm{DDX} \\ \mathrm{DDY} \end{pmatrix} \end{aligned}$$

$$
\begin{aligned}
\|c_p\| &= \sqrt{\mathrm{DX}^2 + \mathrm{DY}^2} \\
&= \mathrm{NORM} \\
N &= J{\cdot}T = J{\cdot}\frac{c_p}{\|c_p\|} = \frac{1}{\mathrm{NORM}}{\cdot}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\cdot\begin{pmatrix} \mathrm{DX} \\ \mathrm{DY} \end{pmatrix} = \frac{1}{\mathrm{NORM}}{\cdot}\begin{pmatrix} \mathrm{DY} \\ -\mathrm{DX} \end{pmatrix} \\
&= \begin{pmatrix} \mathrm{XN} \\ \mathrm{YN} \end{pmatrix} \\
\kappa &= \frac{c_{pp}{\cdot}N}{\|c_p\|^2} = \frac{1}{\mathrm{NORM}^2}\begin{pmatrix} \mathrm{DDX} \\ \mathrm{DDY} \end{pmatrix}\cdot\begin{pmatrix} \mathrm{XN} \\ \mathrm{YN} \end{pmatrix} = \frac{1}{\mathrm{NORM}^2}{\cdot}\,(\mathrm{DDX}{\cdot}\mathrm{XN} + \mathrm{DDY}{\cdot}\mathrm{YN}) \\
&= \mathrm{KAPPA} \\
\|\nabla I(x,y)\|^2 &= \left\|\begin{pmatrix} \frac{I(x+1,y)-I(x\text{-}1,y)}{2} \\ \frac{I(x,y+1)-I(x,y\text{-}1)}{2} \end{pmatrix}\right\|^2 = \left(\frac{I(x{+}1,y) - I(x\text{-}1,y)}{2}\right)^2 + \left(\frac{I(x,y{+}1) - I(x,y\text{-}1)}{2}\right)^2 \\
\phi(x,y) &= \frac{1}{1 + \frac{1}{\mathrm{A}}{\cdot}\,\|\nabla I\|^2} \qquad \text{(A=2000, chosen so that } \phi \text{ looks good (reasonable slope) with } I(x,y) \in \{0...255\}\text{)} \\
&= \mathrm{PHI} \\
\nabla\phi(x,y) &= \begin{pmatrix} \frac{\phi(x+1,y)-\phi(x-1,y)}{2} \\ \frac{\phi(x,y+1)-\phi(x,y-1)}{2} \end{pmatrix} \\
&= \begin{pmatrix} \mathrm{DPHIX} \\ \mathrm{DPHIY} \end{pmatrix} \\
\nabla\phi{\cdot}N &= \mathrm{DPHIX}{\cdot}\mathrm{YN} - \mathrm{DPHIY}{\cdot}\mathrm{XN} \\
&= \mathrm{DPHIN}
\end{aligned}
$$

So finally :

$$
c_i(t+\Delta t) = c_i(t) + \Delta t{\cdot}\,(\mathrm{PHI}{\cdot}\,(\mathrm{KAPPA} + \mathrm{ALPHA}) - \mathrm{DPHIN})\cdot\begin{pmatrix} \mathrm{XN} \\ \mathrm{YN} \end{pmatrix}
$$

3.2 Other implementation considerations

The minimal and maximal distance between particles are defined. Shen two successive particles are closer than the minimal distance, they are merged into one particle in between with linear interpolation, and when two successive particles are farther than the maximal distance, another particle is added in between with linear interpolation. Usually I used 5 and 15 pixels for minimal and maximal distances.

The particles are prevented from going closer than 2 pixels from the image sides, in order to avoid to deal with the image boundaries (this is a simplification because it is not important for the active contour).

The contour is initialized as a circle. The user gives its center, radius, and the direction of propagation (whether it is initialized inside or outside the object).

The algorithm was implemented in C++ with OpenCV. The source code is available here : http://crteknologies.free.fr/publish/pde_snakes-edge-mp.zip.

4 Experimental results

4.1 The coins

I first tested the algorithm with a simple image representing coins on a white background, and initializing the contour outside (figure 1).

FIG. 1 – Test on a simple object with white background (initial and final states)

The time evolution for figure 1 can be seen on the video available here : http://crteknologies.free.fr/publish/pde_coins.avi.

Actually the erosion term is not needed, because the regularity term $\phi{\cdot}\kappa$ already makes the curve shrink if not stopped by $\nabla\phi{\cdot}N$.

However when starting from inside, as there are a lot of details in the coin, the snake is stuck inside with local gradient maxima.

4.2 The pillow

Then I tested with another image from inside the object (figure 2). The pillow is quite uniform inside, but nonetheless has creases on the sides with non negligeable gradient and local gradient maxima, that sometimes slows down the progression because of the stopping term.

Moreover we can notice that the rigidity constraint $\phi{\cdot}\kappa$ prevents it from fitting to the corners.

The time evolution for figure 2 can be seen on the video available here : http://crteknologies.free.fr/publish/pde_pillow.avi.

4.3 The dolphin

Eventually I tested with a more difficult image. The snake didn't converge because it was stuck to details of high gradient (figure 3).

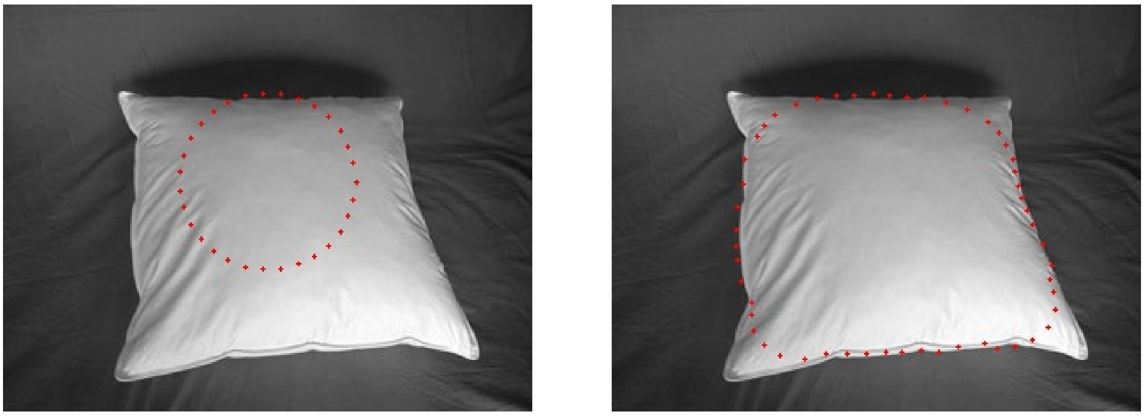

FIG. 2 – Test in a more difficult case (initial and final states

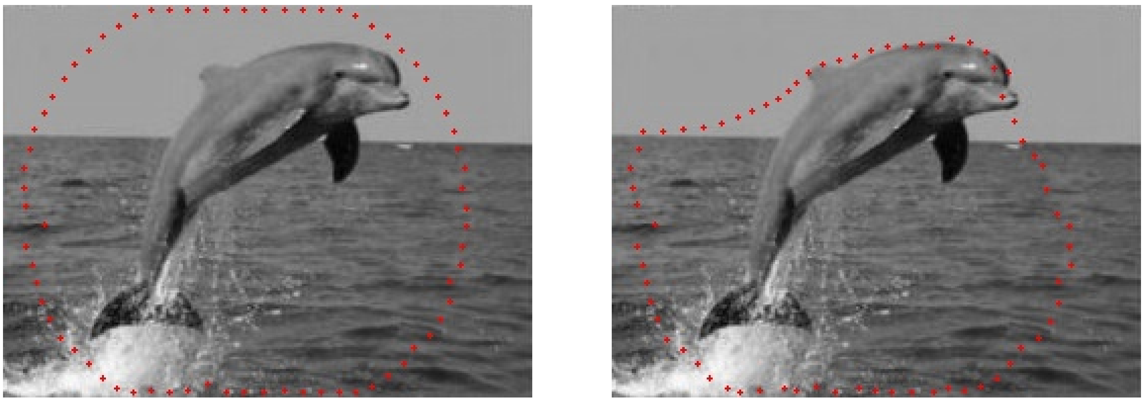

FIG. 3 – Test in a difficult case (initial and final states)

The time evolution for figure 3 can be seen on the video available here : http://crteknologies.free.fr/publish/pde_dolphin1.avi.

However I did several modifications in order to behave better.

First we can see that the active contour is stucked to details of small size, so I smoothed the image with an edge-preserving method (Selective Gaussian function of The Gimp).

Second, we can see that the active contour captures the horizon. This seems normal, but as it is normal to the active contour we can think that it should not be sensitive to this gradient. So I used the scalar product between the gradient of the image and the normal of the contour instead of only the gradient.

Finally, in order to be still less bothered by spurious gradient, I added unconditional terms of regularity and erosion (by unconditional I mean not multiplied by ϕ).

Then it was able to reasonably find the dolphin (figure 4), even if rigidity constraints still prevented it from going in convex parts.

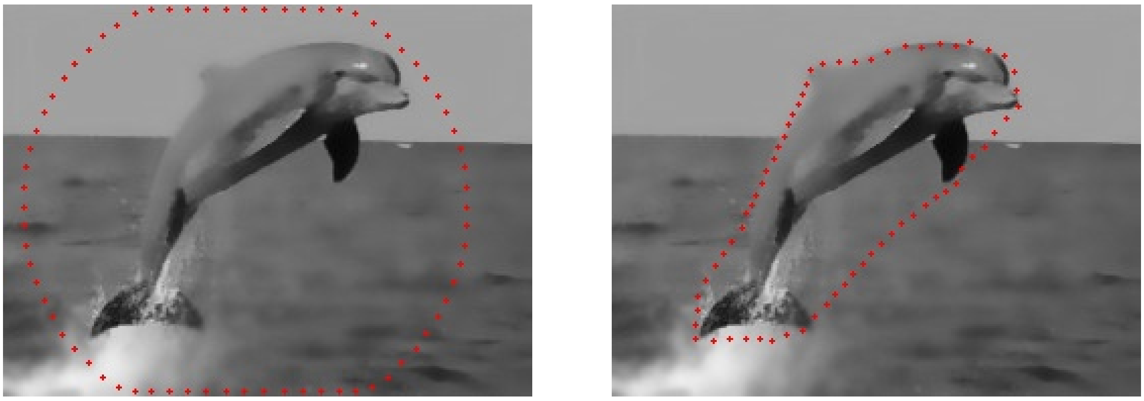

FIG. 4 – Test in a difficult case with modifications (initial and final states)

The time evolution for figure 4 can be seen on the video available here : http://crteknologies.free.fr/publish/pde_dolphin2.avi.

5 Discussion

The main problem I see with this energy functional is that it is pretty sensitive to the strength of the gradient it will stop at. If the object contour has little contrast, the parameters have to be manually tuned to increase the stopping term compared to the propagation (erosion/dilation) term. The ϕ function has an important role in this sensitivity because it determines with which strenght the snake will react to some gradient change. So tuning this function could be interesting.

The active contour sometimes get stuck with some small parasite gradient areas. It could be interesting to smooth a little bit the image before computing the gradient. This could be done with a Gaussian filter of small radius (or in an equivalent way using a Sobel mask to compute the gradient), or even better using a filter that preserves edges (median-filter, anisotropic filtering).

I could also have added a tangential force in order to keep the particles at the same distance from one to another. However as I add and remove particles when they become too far away or too close, it was fine without it.

Some objects contours are not very well defined by a strong gradient. In those cases region-based active contours can be used. It could be interesting to combine edge and region information too.

Moreover implementation with marker particles is not perfect. It is sensitive to initialization, and does not deal with topological changes. Level-set methods are known to be more robust.

To conclude, this implementation of edge-based active contour with marker particles was an interesting first implementation of active contours for me. It has provided me thorough practical understanding that will help me if I have to implement or use other types of active contours or other implementation methods that we have seen the theory in class, or that one can find in the scientific literature.