

AUTOMATIC COMPUTATION OF PARAMETER IN A FAST LEVEL SET METHOD

Ludovic PAULHAC, Julien OLIVIER and Jean-Jacques ROUSSELLE

Université François-Rabelais de Tours

Laboratoire d'Informatique

64, avenue Jean Portalis

37200 Tours, France

ludovic.paulhac@etu.univ-tours.fr, {julien.olivier, rouselle}@univ-tours.fr

ABSTRACT

The *level set* method is included in the implicit active contour family. This algorithm gives good results but it is very slow, that is why we can find a lot of variants like *fast level set*. The problem of *fast level set* we use is that it needs to set a parameter. In this paper we describe a method using an adaptative parameter. For that, this method analyse the gradient in different places in the image to obtain the best parameter according to the local image zone processed. In our experiments, we demonstrate that *fast level set* with an automatic and manual parameter gives similar results concerning contour quality and execution time.

KEY WORDS

Active contour, Fast level set, image processing, adaptative parameter, contour quality

Introduction

Image processing seems easy for a human. Nevertheless, it is difficult to implement it using mathematical and computing tools. For this problem, there are many segmentation methods that we can aggregate in three algorithm main classes depending on whether we want to identify region with its content, with its border, or using global knowledge. The active contours, used in image processing and artificial video, have been introduced by Kass and Witkin in 1987 [1]. They are used for the segmentation of images and particularly to isolate objects in an image. The two active contour families are parametric active contours and implicit active contours. In these two type of active contours, we want to solve this evolution equation:

$$\frac{\partial(C(s,t))}{\partial t} = F_N(s,t).N(s,t) + F_T(s,t).T(s,t) \quad (1)$$

with $C(s,t)$ the evolution curve, $F_N(s,t)$ the evolution speed amplitude of the normal to the curve $N(s,t)$ and $F_T(s,t)$ the evolution speed amplitude of the tangent to the curve $T(s,t)$. The tangent has no effect for the geometric transform of contour. In consequence, we can simplify the evolution equation using only the normal like this:

$$\frac{\partial(C(s,t))}{\partial t} = F_N(s,t).N(s,t) \quad (2)$$

Parametric active contour have been introduced by Kass *et*

al [1] and during these twenty last years, there was many research in this domain. The parametric active contour family is robust and are composed of several methods. In a first time, Kass *et al* [1] suggested a minimization method using technics of variationnel calculation. Amini and al [2] demonstrated that this method is unstable and proposed a new algorithm using dynamic programming. It is more efficient but slower. Moreover, there is a method proposed in [3], called *greedy*, which is more stable and quick. The problem with this method is that it uses a lot of parameters and does not permit topological changes.

Implicit active contour family that includes *level set*, is an other approach and solves these drawbacks. It have been introduced by Osher and Sethian in [4] and developed by Caselles and Maladi [5]. This method describe an object of the dimension n using the dimension $n + 1$. The dimension n is not calculated directly but deducts by the upper function called *level set* function. Indeed, the active contour will not evolve explicitly but with a three dimension function. The evolution of this function will give the one of the contour. We define $\psi : \mathbb{R}^2 \rightarrow \mathbb{R}$ the level set function. This function can be defined like a three dimension function based on the two dimensions of the image and a third dimension the height which gives the different layout levels. So, the contour will be given by the intersection of ψ and the layout define by $\psi = 0$. Initially, the contour is the initial zero level of ψ defined as follow :

$$\psi(C(s,0),0) = 0, \forall s \in [a,b] \quad (3)$$

The relation describe in equation (4) is preserved during ψ evolution :

$$\psi(C(s,t),t) = 0, \forall s \in [a,b], \forall t \in [0,T] \quad (4)$$

Level set moves with an evolution function like this one :

$$\frac{\partial(C(s,t))}{\partial t} = g(|\nabla I|).(v+k).N(s,t) \quad (5)$$

with v a positive constant which define the evolution speed, k the curvature, the $g(|\nabla I|)$ function which tend

toward zero if the gradient takes a great value. Generally, this function is defined as :

$$g(\nabla I) = \frac{1}{1 + (|\nabla I|)^y} \text{ where } y \in \{1, 2\} \quad (6)$$

$N(s, t)$ is the normal of the curve.

Nevertheless, the *level set* method is very slow and need a lot of calculation. For exemple, if the *level set* evolve in a two dimension space, its complexity is $O(n^2)$ with n the number of points of the picture. With dimension three, the complexity is $O(n^3)$. To compare, the *greedy* complexity is $O(n)$. Many algorithms have been proposed to solve this problem of speed like *narrow band* and *fast level set*. The rule of *narrow band* algorithm [6] [7] is to compute ψ function only in a limited zone around the zero level. For each iteration, this algorithm permits to reduce the execution time. With the *fast level set* method [8], calculations are limited by a band composed of two lists (called *Lin* and *Lout* in Figure 1) which are around the evolution curve. This algorithm keeps the same properties than *level set* and improves the speed. But with this method, we have to give an extra parameter. This work propose to compute automatically this parameter.

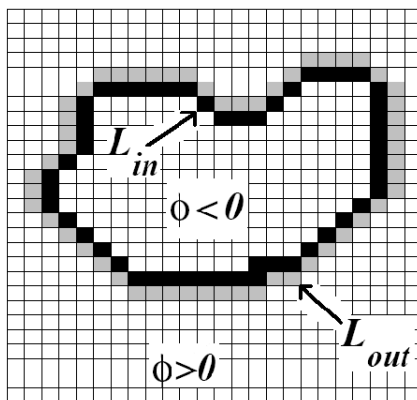


Figure 1. Representation of *Lin* and *Lout*

After to have presented existant *level set*, we will proposed a new method to find automatically the parameter of *fast level set*. Finally, we will compare execution time and contour quality using different methods.

The fast level set parameter

With the *fast level set* method, we use integral values. So the ψ function doesn't use real values which represent distance between pixels and current contour in *level set* method. The ψ function gives only one information which is the location of the evolution contour. Consequently, it is not possible to move the contour using ψ function but only with the speed function. In our speed function we compare the image gradient with a parameter. If the gradient of a contour point is smaller than this parameter, we will return a positive value else a negative value. The problem is to be

able to estimate this parameter. The method we chose is a process which analyse gradient lines of the picture to obtain a parameter as better as possible according to the location in the picture. With these lines we can only keep gradients we need to find the contour.

In order to have an easier segmentation, we chose to apply a processing before using this method. The aim is to intensify boundaries and, to reduce noise we blur in the image. For that, we decided to use the Kuwahara filter [9] discribed in Figure 2.

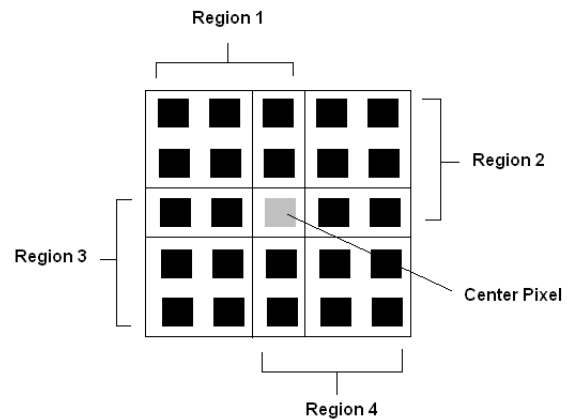


Figure 2. Four regions define for the kuwahara filter. In our case, abscissa = ordinate = 5 and for each region the size is $[(abscissa + 1)/2] \times [(ordinate + 1)/2]$

The central pixel is equal to the gradient mean of the region which has the more little variance. This pixel is set with the mean of the more homogenous region. The image Figure 3 shows the effect of the Kuwahara filter.

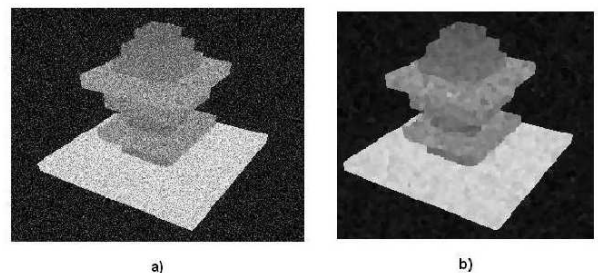


Figure 3. a) An image with noise, b) The same image after a Kuwahara filter

If we compare a gradient line of the original picture with the one obtained with the Kuwahara filter (Figure 4), we can see that filter intensifies gradient picks and reduce noise.

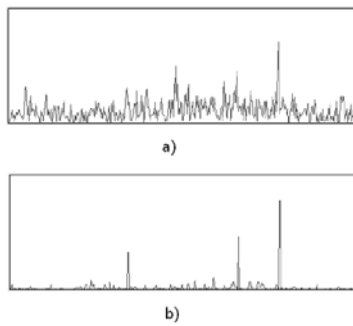


Figure 4. a) Gradient line of the original picture, b) Gradient line after a Kuwahara filter

After this processing we have to find a good parameter. In a first time, we calculated the mean and standard deviation of gradients. To compute a good parameter, we have to find a parameter which permits to keep only the most important gradients. With the mean, we sort many gradients but it is not enough. So we have to add a filter able to keep gradient picks. For that, we use standard deviation which conforms with picture values.

$$coefficient = E(x) + \alpha \times \sqrt{E(x^2) - (E(x))^2} \quad (7)$$

with x the set of pixels of the closest gradient line. After a test on several images, we can see that, if the value of α is upper than 1, then the filter is too-strong and we don't keep gradient picks enough. So, we set α with 1 and like this, we obtain good results. Nevertheless a global coefficient can be a problem if the picture is complexe. That's why we use gradient line to divide the picture in regions and compute a coefficient for each of them. We choose four gradient lines in an image as shown Figure 5.

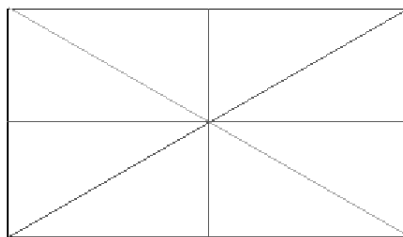


Figure 5. Gradient line initialisation

Using these gradient lines, we can have the position of the contour according to the gradient picks. To know the closest gradient line, we use angle computation. For that, we use the following property : "In a circle, an inscribed angle is the half of the center angle which intercept the same arc." As appear in Figure 6, the angle $\widehat{AOB} = 2\widehat{AMB}$.

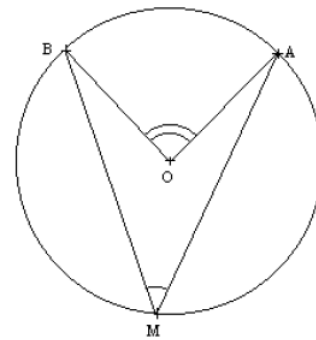


Figure 6. Circle property

If [AM] pass by the center O of the circle, [AM] will be a diameter of the circle and whatsoever the location of point B, the triangle ABM will be a rectangular triangle. Then, we are able to calculate the angle \widehat{AMB} and so the angle \widehat{AOB} .

$$\widehat{AOB} = 2.Asin\left(\frac{chord [AB]}{2.radius}\right) \quad (8)$$

Nevertheless, this method can't find an adapted coefficient when the gradient information is not significative, that is to say, when gradient line doesn't cross an object. Then gradients are constant and the parameter found doesn't give correct results. On the Figure 8-a, we can see that this problem generates many parasite shapes. The line of gradient doesn't cross planes. Consequently, this line is flat because it doesn't contain strong gradients (Figure 7).

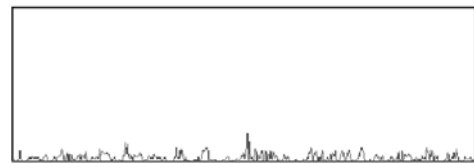


Figure 7. Vertical gradient line

To correct this problem, we chose to compare the mean and the maximum of the gradient. If the maximum gradient is lower than N times the gradient mean, then the parameter is set to "1 + maximum gradient". Like this, the contour will be stopped only by gradient picks in the concerned region. After some tests, we can say that the value seven for N generates good results. Here we try to detect a specific event, that is why N doesn't need a good precision (for exemple ten gives similar results). With this change, we obtain the segmentation shown Figure 8.

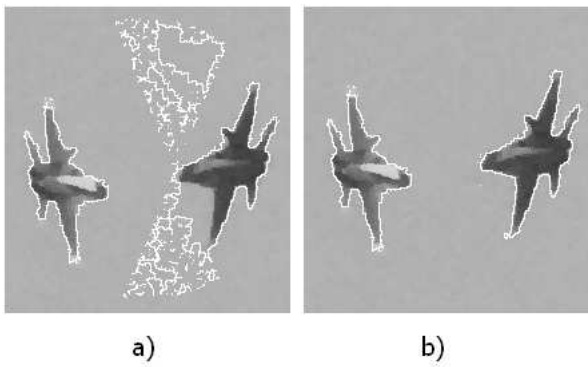


Figure 8. a) The method without a comparison of the mean and the maximum gradient b) The same method with a comparison

Experimental results

In this part, we test the automatic *fast level set* and we compare it with other methods like *Greedy*, *level set*, and manual *fast level set*. To make quality tests, we use Pratt criterion which is a good method [10]. It is an empirique measure which compares a boundary card I_f and a reference card I_{ref} . This method is defined by the following formula:

$$PRA(I_f, I_{ref}) = \frac{1}{MP} \sum_{k=1}^{card(I_f^{cont})} \frac{1}{1 + d(I_f^{cont}, I_{ref}^{cont})} \quad (9)$$

with $MP = \max\{card(I_{ref}^{cont}), card(I_f^{cont})\}$, I_{ref}^{cont} the reference contour and I_f^{cont} the contour tested. This measure return 1 if the expert contour is identical with the test contour. On the contrary, if the test contour is bad Pratt return a value close to 0.

Firstly we test this method with a binary picture (Figure 9).



Figure 9. Binary picture (512 × 384)

With *Greedy* method we need to use specific parameter. We will use the following values with 0.5 for continuity and curvature energy, 1 for balloon energy, 3 for gradient energy, and then 100 points of control. The execution time is 1.3 secondes but the quality is very bad with the value 0.35. Indeed, there are many acute angle in this image and

this is a problem for the *greedy* method.

The *level set* method is more precise than *greedy* with a Pratt quality of 0.992. The problem is the execution time which is too long (1 minute and 40 seconds).

With manual *fast level set* we obtain a good quality with the value 0.992 (with this value, we can say that the expert contour and the test contour are the same). The execution time is 1.55 secondes which is very well.

About automatic *fast level set*, we have the same kind of quality with 0.992. The execution time of automatic *fast level set* and manual *fast level set* is the same. We obtain 2.1 seconds.

Secondly, we test with a noisy image (Figure 10).

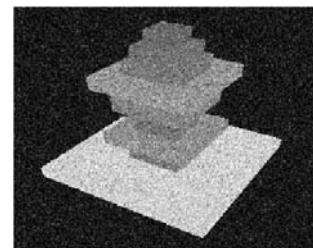


Figure 10. Noisy picture (640 × 480)

The *greedy* (curvature = 0.3, continuity = 0.6, balloon = 1, gradient = 3, point of control = 150) doesn't give a good segmentation. The quality is 0,26 and the execution time is 3 seconds.

Noisy picture is an important problem for *level set* method. Indeed, the contour moves slowly and with many difficulties. That's why execution time is important with this kind of picture. Here, the contour is perturbed by the noise and we obtain the quality 0.

The manual *fast level set* is able to segment this picture with a quality of 0.8 and an execution time of 3.94 seconds.

With automatic *fast level set* the quality is good with 0.73 and the execution time is 4.69 seconds.

To finish, we try with a real image (Figure 11).



Figure 11. Real picture (709×531)

This plane is a complicated shape. With 0.6 for continuity energy, 0.7 for curvature energy, 1 for balloon energy, 4 for gradient energy and with 100 points of control, we obtain the bad quality 0.12 with an execution time of 1.93 seconds.

The *level set* gives a good quality with 0.72. But, we have to wait 18 minutes before the end of the algorithm.

With the manual *fast level set* method the quality is 0.75. Nevertheless, the execution time is much better than *level set* with 2,7 seconds.

The automatic *fast level set* gives similar results with a quality of 0.72 and an execution time of 3 seconds.

To conclude we can say than the results of the automatic *fast level set* are correct. The quality and the execution time are very closed than manual *fast level set*.

Conclusion

We have proposed a new method which permits to compute automatically the parameter of the *fast level set* used here. For that we have seen that a global parameter is not efficient to process all the images. That is why we use gradient lines to divide the image in regions to obtain an adaptative coefficient. Moreover, automatic *fast level set* is a good method because the quality and the execution time is similar with manual *fast level set*. To finish, the advantage of automatic *fast level set* is that we don't waste time trying to trim the parameter. Indeed, it is hard to find the first time the optimal parameter. Nevertheless, we cannot decide which part in the picture we want to segment. The automatic *fast level set* will select the biggest gradients.

References

- [1] M. Kass, A. Witkin, and D. Terzopoulos. Active contour models. *International Journal of Computer Vision*, pages 321–331, 1987.
- [2] A.A Amini, T.E Weymouth, and R.C Jain. Using dynamic programming for solving variational and machine intelligence. *IEEE Transactions on pattern analysis and machine intelligence*, 12(9):855–867, 1990.
- [3] D.J Williams and M. Shah. A fast algorithm for active contours and curvature estimation. *CVGIP : Image Understanding*, 55(1):14–26, January 1992.
- [4] S. Osher and J.A. Sethian. Fronts propagating with curvature dependent speed : Algorithms based on hamilton-jacobi formulations. *journal of computational Physics*, 79:12–49, 1988.
- [5] Vincent Casselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 1(22):61–79, March 1997.
- [6] D. Adalsteinsson and J. Sethian. A fast level set method for propagating interfaces. *journal of computational Physics*, 118:269–277, 1995.
- [7] R. Malladi, J.A Sethian, and B.C Vemuri. Shape modelling with front propagation : A level set approach. *IEEE Transactions on pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.
- [8] Yonggang Shi and William Clem Karl. A fast level set method without solving pdes. *IEEE international conference*, 2:97–100, March 2005.
- [9] M. Kuwahara, K. Hachimura, S. Eiho, and M. Kinoshita. Digital processing of biomedical images. *Plenum Press*, pages 187–203, 1976.
- [10] S. Chabrier, H. Laurent, and C. Rosenberger. Supervised evaluation of synthetic and real contour segmentation results. *European Signal Processing Conference*, 2006.