FURTHER OPTIMIZATIONS FOR THE CHAN-VESE ACTIVE CONTOUR MODEL

Zygmunt L. Szpak and Jules R. Tapamo
School of Computer Science
University of KwaZulu-Natal
Durban, 4062, Republic of South Africa
Email: zygmunt.szpak@cs.ukzn.ac.za and tapamoj@ukzn.ac.za

KEYWORDS

Fast level-set, Optimization, Active Contours Without Edges, Chan-Vese, Shi-Karl, Segmentation

ABSTRACT

When a Chan-Vese active contour model is implemented in a framework that does not solve partial differential equations, we show how the mean pixel intensity inside and outside the curve can be updated efficiently. We reduce each iteration of the Chan-Vese active contour by O(n), when compared to an approach whereby the mean pixel intensities are recalculated for each iteration by looping over the entire image. After implementing the Chan-Vese active contour in the Shi-Karl level-set framework that does not solve partial differential equations, we show that the active contour may be trapped in an idempotent cycle, and we introduce a new stopping criterion to deal with this situation, thereby eliminating wasted computation cycles. Finally, we optimize the regularization cycle in the Shi-Karl framework, by detecting when an additional execution of the regularization cycle has no effect on the active contour, and breaking out of the loop.

1. INTRODUCTION

In (Lakshmanan et al., 2006) and (Pan et al., 2006a), the authors propose efficient implementations for the Chan-Vese *Active Contours Without Edges* model (Chan and Vese, 2001), in a level-set framework that does not require the calculation of partial differential equations. These implementations reduce computation time by using lists of points to represent the contours, and by evolving the contours by adding and removing points from these lists; while concurrently updating a level-set function. Previous work focused on modelling the evolution of the curve, and no mention was made of how to efficiently calculate the mean pixel intensity inside and outside the curve. This is a crucial calculation in the Chan-Vese model. We show how the mean intensity inside the curve and outside the curve can be efficiently updated for each iteration of the curve evolution, and we compare the resulting speed increase against an approach where the mean intensities are recalculated over the entire image at each iteration. Our proposed calculation scheme can be used whenever the mean intensities inside and outside the curve are required, and a list of points representing the curve is used. We implement our solution in the Shi-Karl (Shi and Karl, 2005a) fast level-set framework, and emphasize the speed increase of our method by testing it on large 1024×768 images.

Additionally, we propose a simple modification to the gaussian filtering curve regularization method, used in (Shi and Karl, 2005a), which eliminates redundant cycles and hence decreases the time to convergence. We also show that for certain images, a gaussian filtering regularization method will prevent the active contour from terminating. To overcome this problem we introduce an additional stopping criterion.

The rest of our paper is organized as follows. In Section 2 we summarize the Shi-Karl fast level-set framework, that does not require the calculation of partial differential equations. We review the Chan-Vese piecewise-constant active contour model, and discuss its use and limitations in Section 3. Our proposed optimization for calculating the mean pixel intensity, inside the curve and outside the curve, is presented in Section 4, and our new stopping criterion and optimization for the gaussian filtering regularization method, is introduced in Section 5.

2. SHI-KARL FAST LEVEL-SET METHOD

The level-set method is a numerical technique, for tracking a propagating interface which changes topology over time. It is often used in image segmentation (Paragios and Deriche, 2000). A segmentation is achieved by placing a closed curve on an image, and by evolving the curve according to internal, external and user defined forces. Snakes (Kass et al., 1987) are used in a similar way to segment an image, but in the level-set method, the curve can split and change topology whereas snakes cannot. The traditional level-set method (Sethian, 1999), requires the calculation of partial differential equations, that govern the evolution of the curve. This is a very time consuming calculation. In (Shi and Karl, 2005a), the authors proposed a fast implementation of the level-set method, that does not require the calculation of partial differential equations. Their framework resembles the traditional level-set method, because the curve C is still represented implicitly as the zero level-set of a function ϕ. The function ϕ is defined as the signed distance function, which is positive outside C and negative inside C. From this definition, when a point on C moves inward, its neighboring

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

point that previously was inside the curve C, will lie outside the curve, and so the value of ϕ of that neighboring point will change from negative to positive. Similarly if a point on C moves outwards, the value of ϕ of its outside neighboring point will change from positive to negative. This means that the evolution of the curve C can be controlled without solving partial differential equations, by manipulating the values of ϕ for a list of neighboring points outside C (L_{out}), and a list of neighboring points inside C (L_{in}). Formally, the two lists of neighboring points can be defined as:

$$L_{in} = \{x | \phi(x) < 0 \ \ and \ \ \exists\, y \in N_4(x),\ \phi(y) > 0\} \tag{1}$$

$$L_{out} = \{x | \phi(x) > 0 \ \ and \ \ \exists\, y \in N_4(x),\ \phi(y) < 0\}, \tag{2}$$

where $N_4(x)$ is the discrete 4-connected neighborhood of a pixel x.

To approximate the signed distance function, ϕ is defined as:

$$\phi(x) = \begin{cases} 3, & \text{if } x \text{ is outside } C \text{ and } x \notin L_{Out}; \\ 1, & \text{if } x \in L_{Out}; \\ -1, & \text{if } x \in L_{In}; \\ -3, & \text{if } x \text{ is inside } C \text{ and } x \notin L_{In}. \end{cases} \tag{3}$$

The curve is evolved by switching neighboring pixels between the two lists L_{in} and L_{out}, based on an external speed function F_{ext}, an internal speed function F_{int}, and by updating the level set function ϕ. The external speed is used to attract the curve to the regions of interest, while the internal speed is used to regularize the evolution of the curve so that the curve remains smooth.

The evolution of C is split into two different cycles. In the first cycle, C is evolved according to the external speed; a positive value of F_{ext} moves a point on C outwards (by switching the point from L_{out} to L_{in}), while a negative value of F_{ext} moves a point on C inwards (by switching the point from L_{in} to L_{out}).

The external speed is synthesized from the image and there are many ways to define F_{ext}. For example, the function F_{ext} could be based on the response of an edge detector applied to the image; it could be based on a range of pixel intensities only, or as in the case of the Chan-Vese *Active Contours Without Edges*, the speed could depend on the mean intensities of the regions inside and outside the curve C.

In the second cycle, C is evolved according to the internal speed. The most common way to define F_{int}, is to apply a gaussian filter on the level-set function ϕ for each point on the curve C. The gaussian filter is a weighted sum of a neighborhood of ϕ, centered at a point on the curve C. If the majority of pixels in the neighborhood are inside C, switching a point from L_{out} to L_{in} has a smoothing effect on the curve. Otherwise, if the majority of pixels in the neighborhood are outside C, then switching a point from L_{in} to L_{out} smoothes the curve. This observation can be summarized into the following rule: a point is switched from L_{out} to L_{in} or vice versa, if the sign of ϕ for that point before the gaussian filter is applied to it, is different from the sign of ϕ after the gaussian filter is applied to it.

The evolution of the curve stops when one of two following conditions is satisfied: *(a)* The speeds at each neighboring grid point satisfy:

$$\begin{array}{ll} F(x) \leq 0, & \forall x \in L_{out} \\ F(x) \geq 0, & \forall x \in L_{in}; \end{array} \tag{4}$$

(b) a pre-defined maximum number of iterations N_a, is reached. The pre-defined maximum number of iterations needs to be specified for a noisy image, because the curve may fail to converge to the stable state specified in *(a)*.

This fast level-set framework has been used to solve image processing problems in real time. Before we show how this framework can be used to calculate and update the mean intensities inside and outside the evolving curve, we first review the Chan-Vese model in the next section.

3. CHAN-VESE ACTIVE CONTOUR MODEL

The advantage of the Chan-Vese *Active Contours Without Edges* model, is that it is able to segment an image that has smooth boundaries. It can do this because the evolution of the curve does not depend on gradient information, so weak edges do not affect the final segmentation. The Chan-Vese model suffers from initialization problems (Pan et al., 2006b). The final segmentation is dependent on the placement of the initial curve; sometimes this behavior is desirable

The original formulation of the *Active Contours Without Edges* model (Chan and Vese, 2001), focused on bimodal images. This was later extended to multiphase images (Chan and Vese, 2002). In the bi-modal model, it is assumed that an image I consists of two regions, c_f and c_b, of approximately piecewise-constant distinct intensity values. If the region to be segmented is represented by c_f, then a curve C can be evolved to reach the boundary of c_f by minimizing the energy:

$$F_1(C) + F_2(C), \tag{5}$$

where F_1 and F_2 are defined as follows:

$$\begin{array}{rcl} F_1(C) & = & \displaystyle\int_{inside(C)} |I - c_1|^2 dxdy \\ & \text{and} & \\ F_2(C) & = & \displaystyle\int_{outside(C)} |I - c_2|^2 dxdy. \end{array} \tag{6}$$

C represents the curve, and the variables c_1 and c_2 represent the average intensities inside and outside the curve respectively. If the curve C is inside the region to be segmented, represented by intensity c_f, then $F_1(C) \approx 0$ and $F_2(C) > 0$. If the curve C is outside c_f, then $F_1(C) > 0$ and $F_2(C) \approx 0$. Only when the curve is on the boundary of the region of interest will $F_1 \approx 0$ and $F_2 \approx 0$.

To incorporate this energy minimization into the fast level-set framework, only the external speed function F_{ext} needs to be defined.

3.1 Definition of the External Speed Function

To incorporate the Chan-Vese model into the Shi-Karl level-set framework, the authors in (Lakshmanan et al., 2006) calculate the external speed with:

$$v_i = I(x,y) - c_i \quad i \in \{1,2\}, \tag{7}$$

where $I(x,y)$ is the pixel intensity of a point on the curve, c_1 is the average pixel intensity inside the curve and c_2 is the average pixel intensity outside the curve. The external speed F_{ext} is then defined as:

$$F_{ext} = \begin{cases} 1, & if \;\; v_1 \leq 0 \;\; and \;\; v_2 > 0; \\ -1, & if \;\; v_1 > 0 \;\; and \;\; v_2 \leq 0. \end{cases} \tag{8}$$

This definition of the external speed is not entirely correct. According to this definition, the curve will only expand outwards if the pixel intensity of a point on the curve is greater than the average pixel intensity of the area outside the curve. Figure 1 a) illustrates such an example and the rectangle is successfully segmented. However, by inverting the colors of the test image such that the average pixel intensity of the area outside the curve, is greater than the intensity of a point on the curve, the rectangle is not segmented because the curve fails to evolve outwards (see Figure 1 b)).

To solve this problem we define F_{ext} as:

$$F_{ext} = \begin{cases} 1, & if \;\; |v_1| < |v_2|; \\ -1, & if \;\; |v_1| > |v_2|, \end{cases} \tag{9}$$

where v_1 and v_2 are calculated with equation (7). In the next section we show how the average intensities c_1 and c_2 can be updated efficiently for each evolution iteration of the curve.

4. CALCULATING AND UPDATING THE MEAN PIXEL INTENSITIES

Previous work on fast implementations of the Chan-Vese model (Lakshmanan et al., 2006; Pan et al., 2006a), did not discuss how the mean pixel intensities, outside the curve and inside the curve, can be calculated. We show that the mean intensities c_1 and c_2 can be updated efficiently during the evolution of the curve, since each pixel on the curve in the Shi-Karl fast level-set framework, is moved outward or inwards sequentially, and the intensities of the pixels that are modified in the level-set function ϕ are known. In the traditional level-set framework, where partial differential equations are solved, the level-set function ϕ is evaluated over the entire image, or over a narrow band (Adalsteinsson and Sethian, 1995), and there is no direct knowledge of which pixels have moved outwards, which pixels have moved inwards and which pixels have remained stationary. As a consequence, there is no direct and obvious knowledge of how the mean intensities inside and outside the curve can be updated online. Hence, the mean intensities are calculated by iterating over the entire image, for each iteration of the curve evolution. The original paper on *Active Contours Without Edges* (Chan and Vese, 2001), presents the calculation of the mean intensities as such, and to our knowledge no faster way of calculating the mean intensities has been presented in previous works. In fact, even work that combines the fast Shi-Karl level-set method and the Chan-Vese *Active Contours Without Edges* for real-time contour tracking (Thida et al., 2006), still presents the calculation of the mean intensities inside and outside the curve, as the iteration over the entire image for each frame. For these reasons, we choose the calculation of the mean intensities inside and outside the curve C, by iterating over the entire image for each iteration of the curve evolution, as the baseline against which we compare our proposed method.

Our proposed calculation scheme is simple to implement. Given Ω the image domain, a value v, and Ω_v defined as follows:

$$\Omega_v = \{(x,y) \in \Omega \;/\; \phi(x,y) = v\}, \tag{10}$$

initial values of c_1 and c_2 can be obtained as follows:

$$c_1 = \frac{i_i(-3)}{tp_i(-3)} \tag{11}$$

and

$$c_2 = \frac{i_o(3)}{tp_o(3)}, \tag{12}$$

where for a certain value v, $i_i(v)$, $i_o(v)$, $tp_o(v)$ and $tp_i(v)$ are defined as follows:

$$i_i(v) = \int_{\Omega_v} I(x,y) H(\phi(x,y)) dxdy; \tag{13}$$

$$tp_i(v) = \int_{\Omega_v} H(\phi(x,y)) dxdy; \tag{14}$$

$$i_o(v) = \int_{\Omega_v} I(x,y)(1 - H(\phi(x,y))) dxdy; \tag{15}$$

$$tp_o(v) = \int_{\Omega_v} (1 - H(\phi(x,y))) dxdy; \tag{16}$$

and $H(x)$ is a step function defined as:

$$H(x) = \begin{cases} 1, & \text{if } x > 1; \\ 0, & \text{if } x < -1. \end{cases} \tag{17}$$

In summary, $tp_i(-3)$ is the total number of pixels for which $\phi = -3$; $tp_o(3)$ is the total number of pixels for which $\phi = 3$; $i_i(-3)$ is the sum of intensities of the pixels, for which $\phi = -3$ and $i_o(3)$ is the sum of the intensities of the pixels, for which $\phi = 3$.

Once the initial values for c_1 and c_2 have been calculated, they can be updated efficiently for the evolution of

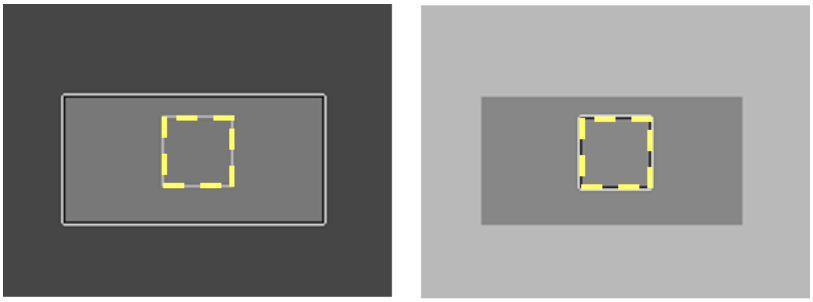

Figure 1: A Poorly Defined External Speed Function **a)** The contour successfully evolves and segments the rectangle. **b)** The contour fails to evolve at all. The initial placement of the curve is overlayed with dashed lines for clarity.

the curve, every time a pixel is switched from L_{in} to L_{out} or vice-versa, and anytime a pixel is removed from a list without placing it in the other list. With this in mind, we modified the two fundamental methods that govern the evolution of the curve in the Shi-Karl fast level-set framework: $switch\_in(x)$ and $switch\_out(x)$.

The purpose of the $switch\_in(x)$ procedure is to move a point on the curve outward by one pixel, while the $switch\_out(x)$ procedure moves a point on the curve inward by one pixel. Our modified procedure $switch\_in(x)$ for a point $x \in L_{Out}$ is defined as:

1. Delete x from L_{Out} and add it to L_{In}. Set $\phi(x) = -1$;
2. $\forall y \in N_4(x)$ satisfying $\phi(y) = 3$, add y to L_{Out} and set $\phi(y) = 1; tp_o \longrightarrow tp_o - 1$ and $i_o \longrightarrow i_o - y$,

while our modified procedure $switch\_out(x)$ for a point $x \in L_{In}$ is defined as:

1. Delete x from L_{In} and add it to L_{Out}. Set $\phi(x) = 1$;
2. $\forall y \in N_4(x)$ satisfying $\phi(y) = -3$, add y to L_{In} and set $\phi(y) = -1$; $tp_i \longrightarrow tp_i - 1$ and $i_i \longrightarrow i_i - y$,

where $N_4(x)$ is the discrete 4-connected neighborhood of x, and i_i, tp_i, i_o and tp_o are defined by equations (13)-(16) respectively.

By adding and subtracting from the variables i_i, tp_i, i_o and tp_o, we can keep track of the necessary information to recalculate the mean intensities c_1 and c_2, without having to iterate over the whole image I.

4.1 Evolving the Curve

Besides $switch\_in(x)$ and $switch\_out(x)$, two more methods $remove\_in(x)$ and $remove\_out(x)$ need to be defined, to explain in detail how the curve C is evolved. The procedure $remove\_in(x)$ is defined as:

1. if $\forall y \in N_4(x), \quad \phi(y) < 0$, delete x from L_{in} and set $\phi(x) = -3$;
2. $tp_i \longrightarrow tp_i + 1$ and $i_i \longrightarrow i_i + x$,

and the procedure $remove\_out(x)$ is defined as:

1. if $\forall y \in N_4(x), \quad \phi(y) > 0$, delete x from L_{out} and set $\phi(x) = 3$;
2. $tp_i \longrightarrow tp_o + 1$ and $i_o \longrightarrow i_o + x$

The purpose of these methods is to remove redundant points that may have been added to L_{in} or L_{out}, when the level-set function ϕ was modified with the $switch\_in(x)$ and $switch\_out(x)$ procedures.

An outline of the algorithm proposed by Shi and Karl, that evolves the curve C, is listed in Algorithm 1. The $\otimes$ symbol denotes convolution, and G is a gaussian kernel. The stopping condition is tested after the curve is evolved according to both the internal force and external force cycles. Some implementations check for the stopping condition immediately after the external force evolution cycle, and skip the internal force evolution cycle if the stopping condition is satisfied. In Section 5, we introduce an additional stopping condition to optimize the gaussian regularization cycle.

4.2 Experimental Results on the Fast Mean Intensity Calculation

We have reduced the time complexity of each iteration of our algorithm by $O(n)$, where n is the number of pixels in image I, by calculating c_1 and c_2 at each iteration based on updated values of i_i, tp_i, i_o and tp_o, instead of calculating c_1 and c_2 by iterating over the entire image I.

To demonstrate the optimization, we ran our algorithm on a large 1024×768 image taken from the Caltech database (Griffin et al., 2007), using an Intel Core 2, 6420 @ 2.13 GHZ with 2 Gigabytes of RAM, and compared it to the algorithm that calculates c_1 and c_2 by iterating over the whole image. We evolved our curve according to the external force F_{ext} only. Refer to Figure 2.

5. OPTIMIZING THE REGULARIZATION METHOD

By evolving a curve according to the external speed F_{ext} only, the curve often develops sharp boundaries due to

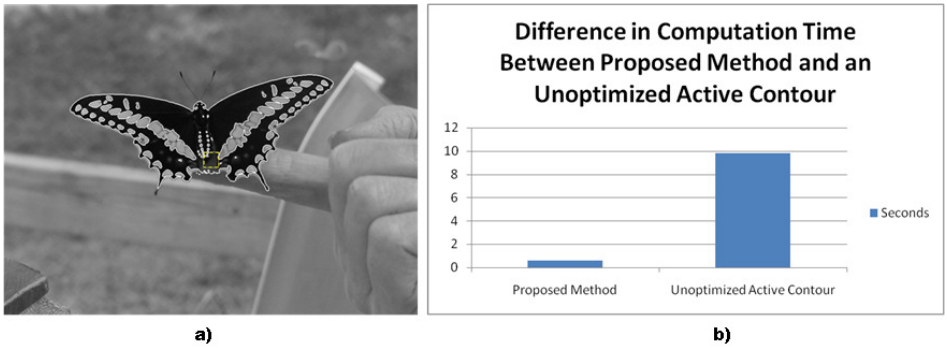

Figure 2: Comparison of Mean Intensity Calculation Methods in Time **a)** The dashed square represents the initial curve boundary. **b)** Difference in computation time between our proposed method, and an implementation in which the mean pixel intensities outside and inside the curve, are calculated by iterating over the entire image for each iteration of the curve evolution. No regularization was used.

Algorithm 1 Algorithm for Curve Evolution in the Shi-Karl Framework

1: Initialize the array ϕ, F, and the two lists L_{in} and L_{out}.
2: **repeat**
3: **for** $i = 0$ to n_{ext} **do** {Evolve according to external speed for n_{ext} iterations}
4: For each point $x \in L_{out}$, $switch\_in(x)$, if $F_{ext} > 0$.
5: For each point $x \in L_{in}$, $remove\_in(x)$.
6: For each point $x \in L_{in}$, $switch\_out(x)$, if $F_{ext} < 0$.
7: For each point $x \in L_{out}$, $remove\_out(x)$.
8: **end for**
9: **for** $i = 0$ to n_{int} **do** {Evolve according to internal speed (smoothness) for n_{int} iterations}
10: For each point $x \in L_{out}$, $switch\_in(x)$, if $(G \otimes \phi)(x) < 0$.
11: For each point $x \in L_{in}$, $remove\_in(x)$.
12: For each point $x \in L_{in}$, $switch\_out(x)$, if $(G \otimes \phi)(x) > 0$.
13: For each point $x \in L_{out}$, $remove\_out(x)$.
14: **end for**
15: **until** equation (4) holds, or N_a iterations have elapsed.

noise. To smooth the curve and to make it less susceptible to noise, an internal speed F_{int} is usually introduced into the model. In the traditional curve evolution methods which are based on partial differential equations, the internal speed is some regularization parameter or function, that is introduced into an energy minimization framework. However, in the Shi-Karl level-set framework that does not solve partial differential equations, the most common approach to smooth the curve is to perform a gaussian filtering on the level-set function ϕ. We summarized the gaussian filter method in Section 2. In this Section we discuss a further optimization to this method and introduce an additional stopping criterion, after demonstrating that performing a gaussian filtering on the level-set function may in some cases prevent the active contour from terminating.

5.1 Gaussian Filtering and Idempotent Active Contours Without Edges

On certain images, the choices of the number of iterations for the external force cycle F_{ext}, and the internal force cycle F_{int}, can cause idempotents. For example, the curve C could be caught in a cycle where it expands outwards due to the external force, and shrinks back to its original shape because of the internal force. This usually happens when the curve has almost reached its optimum boundary, when evolving with the external force F_{ext}, and most pixels are stationary. Refer to Figure 3 for an example of an active contour trapped in such a cycle.

Related work that uses the Shi-Karl fast level-set framework (Shi and Karl, 2005b) and (Thida et al., 2006), presents the stopping criteria a) and b) in equation (4), whereby the parameter N_a is used to ensure the termination of the algorithm for noisy images.

Specifying a pre-determined maximum number of iterations N_a will ensure the termination of the algorithm, but the choice of an appropriate value is difficult. If the chosen value is too low, it can result in a premature termination, and if it is too high it will increase the computation time unnecessarily.

In (Shi and Karl, 2005b), the authors state that when noise in an image is low, one can choose a small value for n_{int} , or increase the parameter n_{ext}, to reduce the percentage of computation allocated for smoothness regularization, thereby speeding up the algorithm (refer to Algorithm 1). This may be true for certain images, but Figure 3 clearly demonstrates that an evolving curve can be trapped in a cycle, even when there is no noise in the image.

We solve this problem by testing for idempotents and stopping the evolution of the curve when a cycle is detected.

After n iterations of evolving the curve according to *both* the external and internal force cycles, the points contained in the outer list L_{out} are defined as $\overline{x}_n$. By evolving the curve C according to the external force (represented by function f; $f^2 = f \circ f$ is the composition of f by itself, and $f^n = \underbrace{f \circ f \circ \ldots \circ f}_{n\ times}$), we have

$$f^{n_{ext}}(\overline{x}_n) = \overline{x}_{n_{ext}}, \tag{18}$$

where $\overline{x}_{n_{ext}}$ are the points contained in L_{out}, after the external force evolution cycle. After evolving according to the internal force (represented by function $\hat{f}$), if we have

$$\hat{f}^{n_{int}}(\overline{x}_{n_{ext}}) = \overline{x}_n, \tag{19}$$

then a cycle has been detected.

Unfortunately, this type of cycle detection involves comparing each element in $\overline{x}_n$ with each element in $\overline{x}_{n-1}$, which is time consuming especially if the lists are not sorted. To avoid making these comparisons, we instead test to see if the number of points in $\overline{x}_n$ is approximately the same as the number of points in $\overline{x}_{n-1}$. In other words, we detect a cycle if

$$|\overline{x}_n| = |\overline{x}_{n-1}| - \epsilon, \text{where } \epsilon \in \mathbb{Z}. \tag{20}$$

We choose ϵ to be a small integer, usually $\epsilon \in \{-2, -1, 0, 1, 2\}$. This is necessary because sometimes the difference between the two lists is only one or two pixels, which have no real impact on the final segmentation; without the ϵ a lot of computation is wasted on insignificant pixels.

5.2 Removing Redundant Cycles During Regularization

We have already mentioned that the number of iterations for the external speed evolution cycle and the internal speed evolution cycle, have to be chosen empirically. Sometimes the number of iterations for the internal speed evolution cycle (regularization) can be set too high, resulting in wasted computation cycles. When the curve C is evolved according to the internal force (represented by function $\hat{f}$), there may be a value of k such that

$$\hat{f}^{n_{int}-k}(\overline{x}_{n_{ext}}) = \overline{x}_n \tag{21}$$

and

$$\hat{f}^{n_{int}-k+1}(\overline{x}_{n_{ext}}) = \overline{x}_n, \tag{22}$$

where $k \in \mathbb{N}$ and n_{int} is the empirically chosen number of iterations for the internal speed evolution cycle. Using equations (21) and (22), we can exit the regularization cycle when we detect that a cycle has not changed $\overline{x}_n$, and avoid unnecessary computations. In practice, to sidestep comparing each element in $\overline{x}_n$ with each element in $\overline{x}_{n-1}$, we instead exit the regularization cycle when equation (20) is true for $\epsilon = 0$.

5.3 Choosing Parameters

In (Shi and Karl, 2005a), the authors mention that the choice of the parameter n_{int}, should normally be set to equal the size N_k of a gaussian kernel. The size of the gaussian kernel (N_k), is geometrically related to the elimination of small holes in the final segmentation; to eliminate holes with a radius smaller than r, $N_k = 2r$. However, in our experiments we found that when n_{int} is greater than the size of the gaussian kernel, segmentation results can sometimes be affected in a positive way (refer to Figure 4 for an example). Hence, we prefer to choose $n_{int} \approx 3N_k$, and allow our algorithm to remove redundant regularization cycles. In this way, the amount of smoothing can vary for each iteration.

The number of iterations devoted to external speed evolution (n_{ext}), is usually greater than n_{int}, since it attracts the curve to the regions of interest.

Finally, the choice of ϵ for the detection of idempotents, is related to the size of the objects in the scene that are to be segmented. For example, if one of the objects is a needle, then it is possible that from one complete curve evolution cycle to the next (using both the external and internal force), the curve expands by only one pixel and so we should choose $\epsilon > 1$. If the objects are large, than it is unlikely that the curve should expand by only one pixel after one complete curve evolution cycle, and so we set $\epsilon \leq 1$.

5.4 Experimental Results on the New Stopping and Regularization Criteria

Figure 5 shows a comparison in running time of an active contour on a test image, with and without our new criteria. The test was conducted on a 640×480 image taken from the Caltech database (Griffin et al., 2007), using an Intel Core 2, 6420 @ 2.13 GHZ with 2 Gigabytes of RAM. The running time of our algorithm was considerably less, because it removed redundant regularization cycles and detected idempotents. The same method of calculating the mean pixel intensities was used throughout the experiment.

In Figure 6 we compare the running time on a large 1600×1600 image of Mars, also taken from the Caltech database. Once again the running time of our algorithm is less. In this image, the difference between the running time is not so great, because the active contour was not trapped in an idempotent cycle, and only 5 iterations were devoted to regularization per evolution cycle. This means that there are at most only 5 redundant regularization cycles, per evolution cycle. By increasing the amount of iterations devoted to regularization, the difference in computation time becomes more noticeable. Nonetheless, even fractions of a second are important in real-time image processing.

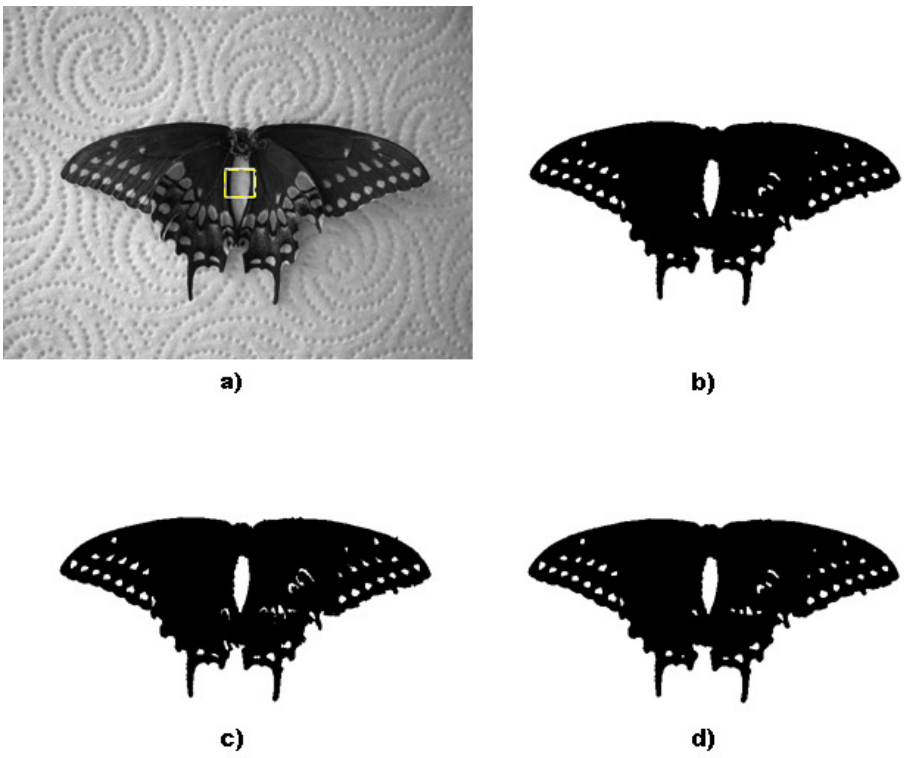

Figure 3: Example of an Active Contour Trapped in an Idempotent Cycle **a)** The initial contour (0 Iterations). **b)** Region inside the curve after 502 iterations. **c)** Region inside the curve after the external speed evolution cycle (522 iterations). **d)** Region inside the curve after the regularization cycle (532 iterations). Notice that **b)** and **d)** are exactly the same. This cycle could continue indefinitely. For example, iteration 562 will be the same as **b)** and **d)**.

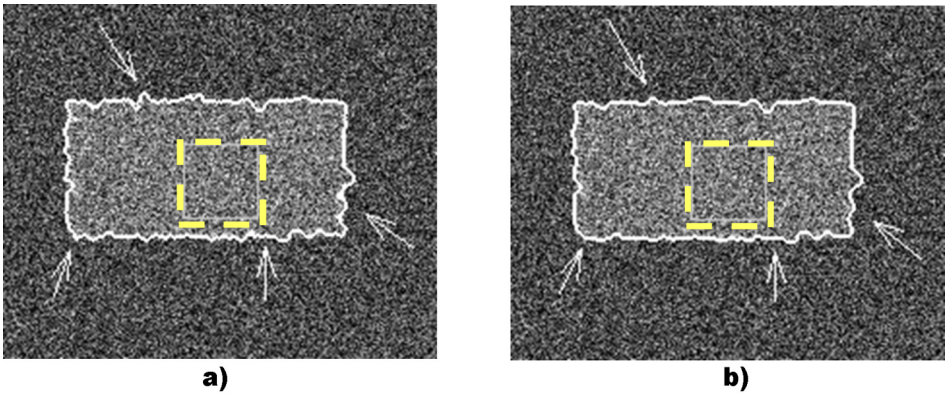

Figure 4: Difference in Regularization Quality **a)** Regularization, by choosing n_{int} as suggested by Shi and Karl (n_{int} equals the size of the kernel). $n_{ext} = 12$, $n_{int} = 5$ and $N_a = 1000$. A 5×5 discrete gaussian kernel was used for regularization. **b)** Regularization, by choosing a large value for n_{int}, and allowing our algorithm to remove redundant regularization cycles. $n_{ext} = 12$, $n_{int} = 10$ and $N_a = 1000$. A 5×5 discrete gaussian kernel was used for regularization. The arrows point to parts of the curve that are smoother. The initial placement of the curve is overlayed with dashed lines for clarity.

6. EXPERIMENTAL RESULTS ON THE COMBINATION OF OUR PROPOSED ALGORITHMS

In Figure 7, we compare both the quality of segmentation and the running time, between an active contour using a standard regularization implementation and using the baseline method of calculating the mean intensities inside and outside the curve, against an active contour using our proposed regularization method and our fast mean intensity calculation scheme. We use a synthetic image of size 512×512, so that a segmentation can be unambiguously evaluated. Clearly, the segmentation result is the same for both methods, while the running time of our method is less.

7. CONCLUSION

We have presented several optimizations for the Chan-Vese *Active Contours Without Edges*, when the active contours are implemented without solving partial differential equations. Using the Shi-Karl framework, we have introduced a fast scheme for updating the mean pixel intensity inside and outside the evolving curve, and we have explained why we chose the baseline for our comparison as the calculation of the mean intensity, by iterating over the entire image for each iteration. Additionally, we have shown that the choice of parameters for the external speed evolution loop and the regularization loop, can trap the active contour in an idempotent cycle, and we have developed a new stopping criterion to break out of that cycle. Finally, we have optimized the regularization loop by exiting the loop, when a further execution of the regularization loop has no impact on the active contour.

REFERENCES

Adalsteinsson, D. and Sethian, J. (1995). "A Fast Level Set Method for Propagating Interfaces". Journal of Computational Physics, No. 118(2), 269-277.

Chan, T. and Vese, L. (2001). "Active Contours Without Edges". IEEE Trans. Image Processing, No.14(10) (Feb.), 266–277.

Chan, T. and Vese, L. (2002). "A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model". International Journal of Computer Vision, No.50(3), 271–293.

Griffin, G.; Holub, A. and Perona, P. (2007). "Caltech-256 Object Category Dataset". California Institute of Technology, 7694, http://authors.library.caltech.edu/7694

Kass, M.; Witkin, A.; and Terzopoulos, D. (1987). "Snakes - Active Contour Models". International Journal of Computer Vision No. 1(4), 321-331.

Lakshmanan, A.; Thida, M.; Chan, K. L. and Zhou, J. (2006). "Incorporation of Active Contour Without Edges in the Fast Level Set Framework for Biomedical Image Segmentation". In *International Conference on Biomedical and Pharmaceutical Engineering* (Singapore, Dec.), 296–300.

Pan, Y.; Birdwell, D. J. and Seddik D. M.(2006). "Efficient Implementation of the Chan-Vese Models Without Solving PDEs". In *Proceedings of International Workshop On Multimedia Signal Processing* (Victoria, BC, Canada, Oct. 03-06), 350–353.

Pan, Y.; Birdwell, D. J.; and Seddik, D. M. (2006). "An Efficient Bottom-up Image Segmentation Method Based on Region Growing, Region Competition and the Mumford Shah Functional". In *Proceedings of International Workshop On Multimedia Signal Processing* (Victoria, BC, Canada, Oct.), 344–348.

Paragios, N. and Deriche, R. (2000). "Coupled Geodesic Active Regions for Image Segmentation: A Level Set Approach". In *Proceedings of ECCV* (Dublin), 224-240.

Sethian, J. (1999). *Level Set Methods and Fast Marching Methods*. Cambridge Monograph on Applied and Computational Mathematics. Cambridge University Press

Shi, Y. and Karl, W. (2005). "A Fast Level Set Method Without Solving Pdes. In *Proceedings of ICASSP* (Philadelphia, PA, USA, Mar.), 97–100.

Shi, Y. and Karl, W. (2005) "Real-time Tracking Using Level-sets". In *Proceedings of CVPR* (San Diego, CA, USA, June) Vol. 2, 34–41.

Thida, M.; Chan, K. L. and Eng, H. L (2006) "An Improved Real-time Contour Tracking Algorithm using Fast Level Set Method". In *Proceedings of the First Pacific Rim Symposium* (Hsinchu, Taiwan, Dec.), 702–711.

AUTHOR BIOGRAPHIES

ZYGMUNT L. SZPAK is a Master's student at the School of Computer Science at the University of KwaZulu-Natal, South Africa. His general research interests include Artificial Intelligence, Image Processing, Computer Vision and Pattern Recognition. Currently, the central theme of his research is on real-time tracking and modelling of the behavior of ships, in a maritime environment. His email is `zygmunt.szpak@gmail.com`.

JULES R. TAPAMO is Associate Professor at the School of Computer Science at the University of KwaZulu-Natal, South Africa. He completed his PhD degree from the University of Rouen (France) in 1992. His research interests are in Image Processing, Computer Vision, Machine Learning, Algorithms and Biometrics. He is a member of the IEEE Computer Society, IEEE Signal Processing Society and the ACM. He maintains a Computer Vision, Image Processing and Data Mining research webpage at `http://www.cs.ukzn.ac.za/cvdm`. His email is `tapamoj@ukzn.ac.za`.

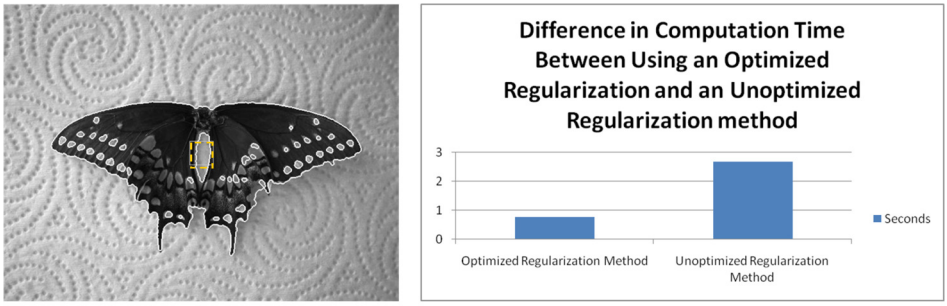

Figure 5: Comparison of Regularization Methods in Time **a)** The dashed square represents the initial curve boundary. **b)** Difference in computation time between our proposed optimized regularization method, and a standard regularization implementation that does not remove redundant regularization cycles, and that does not check for idempotents. $n_{ext} = 10$, $n_{int} = 5$ and $N_a = 1000$. A 5×5 discrete gaussian kernel was used for regularization.

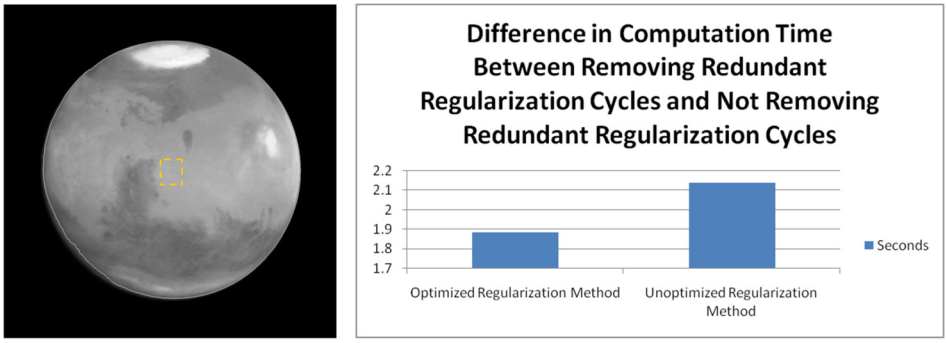

Figure 6: Comparison of Regularization Methods in Time **a)** The dashed square represents the initial curve boundary. **b)** Difference in computation time between our proposed optimized regularization method, and a standard regularization implementation that does not remove redundant regularization cycles, and that does not check for idempotents. $n_{ext} = 10$, $n_{int} = 5$ and $N_a = 2000$. A 5×5 discrete gaussian kernel was used for regularization.

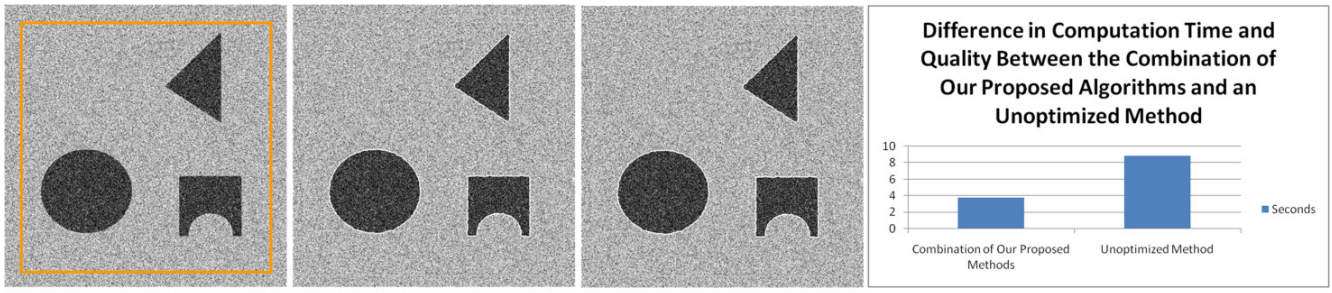

Figure 7: Comparison of the Combination of our Proposed Optimizations against an Unoptimized Method in Time and Quality **a)** Synthetic image corrupted by gaussian noise, with $\mu = 0$ and $\sigma = 80$. The square represents the initial curve boundary. **b)** Segmentation result of our proposed regularization methods, together with our fast mean intensity calculation scheme. **c)** Segmentation result by calculating the mean intensities inside and outside the curve, for each iteration, and using a standard regularization implementation that does not remove redundant cycles nor check for idempotents. **d)** Difference in computation time between the combination of our proposed methods (b), and an approach that calculates the mean intensities inside and outside the curve, for each iteration, and uses a standard (Shi-Karl) regularization implementation, that does not remove redundant regularization cycles, and that does not check for idempotents (c). $n_{ext} = 10$, $n_{int} = 5$ and $N_a = 1000$. A 5×5 discrete gaussian kernel was used for regularization.