Unsupervised Non-parametric Region Segmentation Using Level Sets

Timor Kadir Michael Brady
Department of Engineering Science
University of Oxford
Ewert House, Ewert Place
Summertown, Oxford
OX2 7DD. UK
timork, jmb@robots.ox.ac.uk

Abstract

We present a novel non-parametric unsupervised segmentation algorithm based on Region Competition [21]; but implemented within a Level Sets framework [11]. The key novelty of the algorithm is that it can solve $N \geq 2$ class segmentation problems using just one embedded surface; this is achieved by controlling the merging and splitting behaviour of the level sets according to a Minimum Description Length (MDL) [6, 14] cost function. This is in contrast to N class region-based Level Set segmentation methods to date which operate by evolving multiple coupled embedded surfaces in parallel [3, 13, 20]. Furthermore, it operates in an unsupervised manner; it is necessary neither to specify the value of N nor the class models a-priori.

We argue that the Level Sets methodology provides a more convenient framework for the implementation of the Region Competition algorithm, which is conventionally implemented using region membership arrays due to the lack of a intrinsic curve representation. Finally, we generalise the Gaussian region model used in standard Region Competition to the non-parametric case. The region boundary motion and merge equations become simple expressions containing cross-entropy and entropy terms.

1. Introduction

Despite much effort and significant progress in recent years [6, 8, 10, 16, 21] image segmentation remains a notoriously challenging computer vision problem. Notwithstanding Marr's contention that segmentation is poorly defined as a standalone task [9], there are many applications for which the partitioning of an image into a tessellation of non-overlapping regions, each defined by an appropriate homogeneity criterion, is a useful process. Perhaps driven by the need to address this restricted definition of the problem, numerous approaches have been proposed.

Region Competition [21] attempts to unify some the disparate approaches taken to segmentation. One of its main advantages is that it operates in an entirely unsupervised manner; it is not necessary to specify the classes nor their number prior to segmentation. It solves the segmentation problem using both feature-space and spatial information simultaneously. This is particularly important for textured images where region classes can be difficult to distinguish using feature-space properties alone. However, one difficulty associated with Region Competition is in its implementation. Typically, a pixel region-membership representation is employed. Such a representation lacks an intrinsic curve model which necessitates explicit handling of situations where multiple regions meet and compete. Furthermore, the original formulation given in the paper uses Gaussian region models which can be problematic, in particular for images with textured regions.

Recently, the Level Sets methodology [11] has received a great deal of attention within the Vision community, especially as a framework for segmentation type problems [3, 13, 20]. Originally proposed as a form of Active Contour using only edge information [2], the approach has now been developed to include region information [13]. One of the key attractions of the Level Sets approach is its ability to handle changes in curve topology in a natural way. However, conventional approaches to deal with multiple class segmentation tasks require N coupled surfaces, or at best log N such surfaces [3], to represent N classes [20]. This is not only computationally inefficient but, since these surfaces must be set-up prior to surface evolution, an initial step is required to estimate the desired number of classes (or for [3] the maximum desired number of classes).

In this paper, we present a novel unsupervised segmentation algorithm using a Region Competition algorithm implemented within a Level Sets framework. Our method

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

solves $N \geq 2$ -class problems on one embedded surface and can operate in an unsupervised manner, that is, neither the number nor the form of the partition classes need to be specified a-priori. We contend that the Level Sets methodology provides a more convenient framework for the implementation of the Region Competition algorithm than region membership arrays. Level Sets implicitly define multiple curve fronts, thereby handling region merging in an intrinsic manner. Furthermore, region membership can be directly obtained from the evolving surface. Finally, we generalise the region models used in standard Region Competition, which are Gaussian, to the non-parametric case. The region boundary motion and merge equations become simple expressions containing cross-entropy and entropy terms.

1.1. Region Competition

Here, we provide a brief overview of the Region Competition algorithm. Further details may be found in [21]. Region Competition is an MDL [6, 14] like segmentation algorithm but with a simplified image model and a different cost minimisation algorithm. Whereas Leclerc [6] developed several different MDL image models, in the original paper Region Competition concentrates solely on the piecewise constant case; images are modelled as comprising of regions whereby each region pixel is considered as a sample taken from a single PDF.

The image model comprises three elements. The first is the region model itself, which, in the original formulation, region pixels are modelled as samples from a PDF. The second part of the model is the overhead cost associated with the region model itself. Increasing this encourages fewer regions to be formed. The third part of the model is the boundary cost; the cost associated with coding the length of a region's boundary. This is equivalent to the curvature term in Active Contours [21].

A two-stage iterative algorithm is used to (locally) minimise the global cost in an EM-like manner. The method is initialised by placing 'seeds' across the image. In the first stage, COMPETE, the number of regions is fixed and their boundaries are adjusted as to minimise the global cost. The second stage, MERGE, is to consider the merging of regions by measuring the change in global cost if two adjacent regions are merged. In essence, the purpose of this operation is to overcome local minima in the search space. For example, the local boundary competition might arrive at a steady state between two regions, whereas a lower global cost could be achieved through merging the regions.

2. Non-Parametric Region Competition

The standard Region Competition cost function, in fact most statistical region-based segmentation methods, typically assume parametric PDFs for modelling the regions, often a Gaussian. The reasons are mainly mathematical tractability. However, we have found that the use of such models makes the algorithm very sensitive to values of the initial seed size, local scale and merge cost parameters especially where the region statistics diverge from the Gaussian case. Li et al [7] have proposed the Wilcoxon W-statistic as a means of generalizing Region Competition and more recently Tu and Zhu [17] have proposed a non-parametric segmentation method using MCMC search. Here, we derive a non-parametric region model directly from the MDL cost function.

The general expression for the statistical region force acting on a point $\mathbf{x} = (x, y)$ from a region R_i [21] is:

$$\mathcal{F}_{R_i}(x,y) = \frac{1}{m} \iint_{\mathcal{W}_{(x,y)}} \log P(I_{(u,v)}|\alpha_i) du dv \qquad (1)$$

where $I_{(u,v)} \in D = \{d_1, d_2, \ldots, d_r\}$ and m is the size of the local window, $\mathcal{W}$ taken around $\mathbf{x}$ and α_i are the parameters of the statistics in region R_i. Equation 1 measures the average probability of the pixels in window $\mathcal{W}$ conditional on the parameters of the PDF of region R_i. Therefore we can rewrite (1) as sum over the pixels $(1..m)$ in $\mathcal{W}$:

$$\begin{aligned} \mathcal{F}_{R_i}(x,y) &= \frac{1}{m} \sum_{j=1}^{m} \log\left(p_{R_i}(I_j)\right) \qquad\qquad I_j \in \mathcal{W}_{(x,y)} \\ &= \frac{1}{m} \left[\log\left(p_{R_i}(I_1)\right) + \ldots + \log\left(p_{R_i}(I_m)\right)\right] \qquad (2) \end{aligned}$$

Next, defining the counting function $\mathcal{C}(s) = \sum_j \delta_{I_j s}$ where $\delta_{pq} =$ the Kronecker delta, we can rewrite (2) as a sum over the descriptor values D as follows:

$$\begin{aligned} \mathcal{F}_{R_i}(x,y) &= \frac{1}{m}[\mathcal{C}(d_1)\log\left(p_{R_i}(d_1)\right) + \ldots \\ &\qquad + \mathcal{C}(d_r)\log\left(p_{R_i}(d_r)\right)] \\ &= \sum_{d \in D} p_{\mathcal{W}_{(x,y)}}(d) \log\left(p_{R_i}(d)\right) \qquad (3) \end{aligned}$$

where $p_{\mathcal{W}_{(x,y)}}(d)$ and $p_{R_i}(d)$ are the PDFs of the image in a local window $\mathcal{W}$ around $\mathbf{x}$ and the region R_i respectively. We may recognise that Equation 3 is the (negative) cross-entropy between these regions. Similarly, we derive an expression for the merge cost:

$$\begin{aligned} \Delta E_M = \; & -\lambda + n_{\mathcal{R}_i} \sum_{d \in D} p_{\mathcal{R}_i}(d) \log p_{\mathcal{R}_i}(d) + \\ & n_{\mathcal{R}_j} \sum_{d \in D} p_{\mathcal{R}_j}(d) \log p_{\mathcal{R}_j}(d) \\ & -n_{\mathcal{R}_{ij}} \sum_{d \in D} p_{\mathcal{R}_{ij}}(d) \log p_{\mathcal{R}_{ij}}(d) \qquad (4) \end{aligned}$$

where $p_{R_i}(d)$, $p_{R_j}(d)$, $p_{R_{ij}}(d)$ and $n_{\mathcal{R}_i}$, $n_{\mathcal{R}_j}$ and $n_{\mathcal{R}_{ij}}$ are the PDFs and numbers of pixels inside regions $\mathcal{R}_i$, $\mathcal{R}_i$ and

$\mathcal{R}_{ij}$ respectively. We may recognise that the region parts of Equation 4 are simply the (negative) entropies of pixels within each region. In our implementation normalised histograms are used to approximate the local region PDFs.

3. Level Set Region Competition

Where the number of classes is greater than two, segmentation algorithms implemented within a Level Set framework have used multiple coupled surfaces to date. Such an approach is difficult to adopt here since Region Competition demands that regions be created and destroyed dynamically. We note that the problem is essentially one of representation; the conventional single surface Level Set approach can represent only two classes: positive and negative regions of the surface ψ. Therefore we need a solution whereby disconnected regions are considered separately. Our solution is to use a connected region labelling algorithm at each iteration of the curve evolution to extract the separate regions.

The only problem that remains is to control the merging of the regions such that merging is only allowed if the global cost is reduced. Our solution to this is to detect when two or more disconnected regions of the same polarity come near one another[1] and depending on the outcome of a merge cost calculation, allow them to merge or cause them to compete over the boundary pixels. This condition can be detected by conducting a local neighbourhood search for pixels of the same polarity but differing region label at each boundary point in the current surface function.

Although the proposed approach is straightforward in principle, it requires some careful definitions and coding. We detail these in the following sections. The main iteration part of the algorithm (i.e. this is applied after initialisation) is given in Section 4 in pseudo-code. The pseudo-code is provided to give a concise and complete explanation of the proposed algorithm, but the principles of our approach may be implemented in many different ways. That having been said, it should be noted that the order of the operations is important in the pseudo-code.

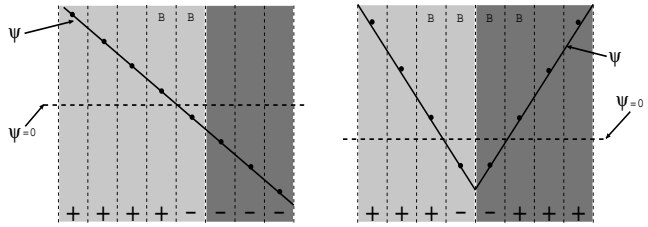
3.1. Evolution Equation

The update equation used in our implementation is:

$$\psi = \psi - \mathcal{F}_{Region}|\nabla\psi| * \mathrm{dt} \tag{5}$$

where $\mathcal{F}_{Region}$ is the speed function defined in Section 3.4 and dt is the time step used in the discrete derivative, typically set to 0.4. Equation 5 is implemented using conventional Level Set finite difference approximations [15]. We also employ the Narrow Band optimisation technique [1] in our implementation. We have not include the conventional curvature dependent term. If required, it can be included as an extra term as part of the cost functions.

[1] In practise, due to the manner in which regions are defined we need only detect one of the same polarity cases; i.e. either positive/positive or negative/negative. In our implementation we choose to detect the positive/positive case.



(a) FG/BG Interface (b) FG/FG Interface

Figure 1. Illustrative example depicting two regions in the proposed scheme. The vertical dash lines represent pixel boundaries. Boundary pixels ('B') for each region include pixels in the zero level set and the adjacent negative pixels. Hence, two negative pixels must separate each positive region.

3.2. Region and Boundary Definition

We define three main elements, Boundaries (B), Foreground (FG) and Background (BG), as follows:
Boundaries. The zero level set of ψ, or in practise the nearest positive and negative pixels. Marked 'B' in Figure 1.
Foreground Regions. The positive regions of ψ but include the boundary pixels. In Figure 1(a) the light grey region is the foreground and in Figure 1(b) both the dark and light grey regions are foreground.
Background Regions. These are defined as the negative regions of ψ excluding any pixels assigned to foreground. In Figure 1(a) the dark grey region is the background.

The use of these unconventional region and boundary definitions simplifies the implementation of the region control logic. For example, with these definitions the foreground regions can both grow and shrink; to grow outwards a positive region has to increase above zero those pixels beyond its positive boundary and to shrink it must reduce below zero those pixels at the edge of its positive boundary.

Initialisation is carried out in the conventional manner; seeds are placed either on a grid, randomly or as a result of some pre-processing step. In our implementation, seeds are defined as foreground regions. Direct evaluation of the signed distance function or a fast-marching method may be used to generate ψ.

In order to obtain the pixel memberships of different regions from ψ, a modified region labelling algorithm is used. This is step 01 in Section 4. Two modifications are made.

The first, is to label foreground regions with positive labels and background regions with negative labels, and the second is to label as foreground those negative pixels immediately adjacent to foreground regions. The first, although not strictly necessary, facilitates a simpler algorithm description and implementation. The second is necessary for the definitions of regions.

Boundary pixels are found using a standard zero crossing detector modified such that both positive and negative pixels are classified as boundaries. This is step 02 in Section 4. Speeds are set only at or in the immediate vicinity of boundary pixels and most of the region control logic operates on and around the boundary pixels. For example, see steps 03, 07 and 21 in Section 4.

It should be noted that despite the above definitions, both types of region are treated in an identical manner by the cost function; the distinction is necessary only for the different COMPETE and MERGE operations for each case. These are discussed next.

3.3. Merging and Competition

For the COMPETE part of the algorithm, the change in energy at a particular pixel, $\Delta E_{(x,y)}$ is calculated, for the Gaussian region model, by:

$$\Delta E_{(x,y)} = \log \frac{\sigma_i^2}{\sigma_j^2} + \left(\frac{(\bar{I}_{(x,y)} - \mu_i)^2}{\sigma_i^2} - \frac{(\bar{I}_{(x,y)} - \mu_j)^2}{\sigma_j^2} \right) + \left(\frac{S^2_{(x,y)}}{\sigma_i^2} - \frac{S^2_{(x,y)}}{\sigma_j^2} \right) \quad (6)$$

where μ_i, σ_i^2 are the mean and variance for region i; μ_j, σ_j^2 the mean and variance for region j; and $\bar{I}_{(x,y)}, S_{(x,y)}$ are the mean and variance in a local window around a boundary pixel (x, y). For the non-parametric region model case, $\Delta E_{(x,y)}$ is given by:

$$\Delta E_{(x,y)} = \sum_{d \in D} p_{\mathcal{W}_{(x,y)}}(d) \log \left(p_{R_j}(d) \right) - \sum_{d \in D} p_{\mathcal{W}_{(x,y)}}(d) \log \left(p_{R_i}(d) \right) \quad (7)$$

where $p_{\mathcal{W}_{(x,y)}}(d)$ represents the PDF in a local region around the boundary pixel (x, y), and $p_{R_i}(d), p_{R_j}(d)$ are the PDFs within region i and region j respectively.

The MERGE part of the algorithm evaluates potential merges by calculating the change in global cost, ΔEM. For the Gaussian region model, this is given by:

$$\Delta EM_{(i,j)} = -\lambda + \frac{1}{2} \left(n_{\mathcal{R}_i} \log \frac{\sigma_{ij}^2}{\sigma_i^2} + n_{\mathcal{R}_j} \log \frac{\sigma_{ij}^2}{\sigma_j^2} + 1 \right) \quad (8)$$

where λ is the overhead cost in encoding a region; σ_i^2, σ_j^2 and σ_{ij}^2 are the variances for regions i, j and the union of regions i and j respectively; and $n_{\mathcal{R}_i}, n_{\mathcal{R}_j}$ are the numbers of pixels in regions i, j respectively.

For the non-parametric region model, ΔEM is given by:

$$\Delta EM_{(i,j)} = -\lambda + n_{\mathcal{R}_i} \sum_{d \in D} p_{\mathcal{R}_i}(d) \log p_{\mathcal{R}_i}(d) + n_{\mathcal{R}_j} \sum_{d \in D} p_{\mathcal{R}_j}(d) \log p_{\mathcal{R}_j}(d) - n_{\mathcal{R}_{ij}} \sum_{d \in D} p_{\mathcal{R}_{ij}}(d) \log p_{\mathcal{R}_{ij}}(d) \quad (9)$$

where λ is the overhead cost in encoding a region; $p_{\mathcal{R}_i}, p_{\mathcal{R}_j}$ and $p_{\mathcal{R}_{ij}}$ are the PDFs for regions i, j and the union of regions i and j respectively; and $n_{\mathcal{R}_i}, n_{\mathcal{R}_j}$ and $n_{\mathcal{R}_{ij}}$ are the numbers of pixels in regions i, j and the union of regions i and j respectively.

An important point to note is that merges must be restricted such that only unique pairs of regions can merge at each step. The problem is that due to the local way in which merges are tested, two regions might be incorrectly merged via a third adjacent region. In our implementation this is done by setting a flag for each region indicating pending merges. The flag, `MFlag(label)` is set to the label number of the region with which it is about to merge and zero otherwise. In Section 4, steps 38-39 set this flag and 25-26 test it.

3.4. Region Control Logic

The two stages of COMPETE and MERGE require different operations depending on whether adjacent regions are foreground or background. In order to produce the correct behaviour, it is first necessary to detect whether nearby regions are foreground or background. This can be done by conducting a local neighbourhood search at each boundary pixel; an 8- or 16-neighbourhood can be used. Foreground regions are present where any of the pixels in the local search area have a **positive label** and this label differs from that of the current boundary pixel. Conversely, background regions are present where any of the pixels in the local search area have a **negative label**. See steps 07-09 and 21-23 in Section 4 for the pseudo-code definitions for these tests respectively. The appropriate actions for each of these combinations is specified below.

Foreground vs. Background. The operation in COMPETE mode for this case is straightforward. For each member in the set of boundary pixels the front speed is calculated according to:

$$\mathcal{F}_{Region}(x, y) = \begin{cases} -=1 & \text{if } \Delta E_{(x,y)} <= 0 \\ +=1 & \text{if } \Delta E_{(x,y)} > 0 \end{cases} \quad (10)$$

where $-=1$ and $+=1$ indicate decrement and increment operations. For simplicity we have used a binary speed function. A smoother speed function would result in improved convergence and final segmentation. Investigation of alternative speed functions is ongoing.

The MERGE operation operates by 'overriding' the speed set by the Foreground vs. Background COMPETE

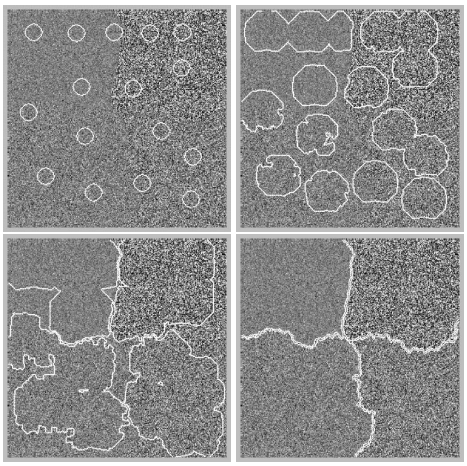


Figure 2. Segmentation of a synthetic image composed of 4 regions containing Gaussian noise with identical means (128) and different variances (0.01, 0,02, 0.04, 0.08). Gaussian region model.

if it causes the overall cost to reduce. Potential merge operations of adjacent regions are tested at each iteration by calculating the change in global cost if the merge were to take place. If the global cost is reduced then the speed for the foreground region is set to -1 (i.e.'grow'), otherwise the speeds from the COMPETE calculations are used:

$$\mathcal{F}_{Region}(x,y) = \begin{cases} -1 & \text{if } \Delta EM_{(i,j)} < 0 \\ COMPETE & \text{if } \Delta EM_{(i,j)} >= 0 \end{cases} \quad (11)$$

Equation 11 gives the speed at boundary pixel (x, y) which separates regions i and j.

Foreground vs. Foreground. For the COMPETE operation in this case, we would like one region to push the other away if it causes the global cost to decrease. Equations 6 or 7 for the Gaussian and non-parametric models respectively, determine which of any adjacent Foreground regions should grow and which should shrink.

To effect the required behaviour, our approach is to cause the region(s) with the weaker statistical force, that is the region whose statistics are less similar to those of the local region around the boundary position under consideration, to shrink (speed=1); stronger regions are not moved. This causes a small background region to appear between adjacent regions and at the next iteration the stronger will move into that area in the normal Foreground vs. Background COMPETE operation. This is done in steps 33-34 in Section 4.

For the MERGE operation, the merge criterion (Equations 8 or 9) is tested. If the global energy is reduced by allowing two regions to merge then nothing needs to be done; the normal Foreground vs. Background COMPETE will cause the two to merge.

3.5. Cleaning up old boundaries

Due to their definition, boundary pixels from foreground regions that previously merged remain in the boundary map. This does affect the region map but does result in a confusing the boundary map. We can correct this behaviour by setting the default speed of all boundary pixels that meet other boundary pixels from the same region ($label <= 0$ and $label_{FG1} = label_{FG2}$) to grow (speed=-1).

4. The Algorithm

In this section, we provide an pseudo-code description of one iteration in the proposed algorithm.

```
01  Label=Find all region labels
02  BoundaryList=Find all boundary pixels
03  For each boundary pixel (x,y) in BoundaryList do:   /* Grow Speeds */
04    Speed(x,y)=-1
05    For each pixel (u,v) in an 8 neighbourhood around (x,y) do:
06      If (Label(x,y)!=Label(u,v)) then Speed(x,y)=0
07  For each boundary pixel (x,y) in BoundaryList do:   /* FG vs. BG */
08    For each pixel (u,v) in an 8-neighbourhood around (x,y) do:
09      If (Label(x,y)!=Label(u,v) AND Label(u,v)<0) then:
10        {
11        DELTA_EM=CalcMergeCost(Label(x,y), Label(u,v))
12        If (DELTA_EM>=0) then:                        /* COMPETE */
13          {
14          LStats=CalcLocalStats(x,y)
15          DELTA_E=CalcCoherence(LStats, Label(x,y), Label(u,v))
16          If (DELTA_E>0) then Speed(x,y)+=1
17          Else Speed(x,y)-=1
18          }
19        Else Speed(x,y)=-1                            /* MERGE */
20        }
21  For each boundary pixel (x,y) in BoundaryList do:  /* FG vs. FG */
22    For each pixel (u,v) in an 8-neighbourhood around (x,y) do:
23      If (Label(x,y)!=Label(u,v) AND Label(u,v)>0) then:
24        {
25        DELTA_EM=CalcMergeCost(Label(x,y), Label(u,v))
26        If (DELTA_EM>=0 OR ( MFlag(Label(x,y))>0
27            AND MFlag(Label(x,y))!=Label(u,v) )) then:  /* COMPETE */
28          {
29          Speed(x,y)=0
30          LStats=CalcLocalStats(x,y)
31          DELTA_E=CalcCoherence(LStats, Label(x,y), Label(u,v))
32          If (DELTA_E>0) then
33            For each pixel (s,t) in an 8-neighbourhood around (x,y) do:
34              If (Label(s,t)=Label(x,y)) then Speed(s,t)=1
35          }
36        Else                                          /* MERGE */
37          {
38          MFlag(Label(x,y))=Label(u,v)
39          MFlag(Label(u,v))=Label(x,y)
40          }
41        }
```

`CalcMergeCost` and `CalcCoherence` are Equations 8 or 9 and 6 or 7 respectively.

5. Results

In the first experiment, the algorithm was applied to an image comprising regions with identical means but with different variances; a similar test image was used in [21]. Figure 2 shows the results. This example is useful because it demonstrates that the competition part of the algorithm works correctly. In the initialisation, we deliberately placed a seed point on the boundary between two image regions[2].

[2]We use the term image region to mean the regions of the image and segmentation region to mean the regions defined by the Level Set function.

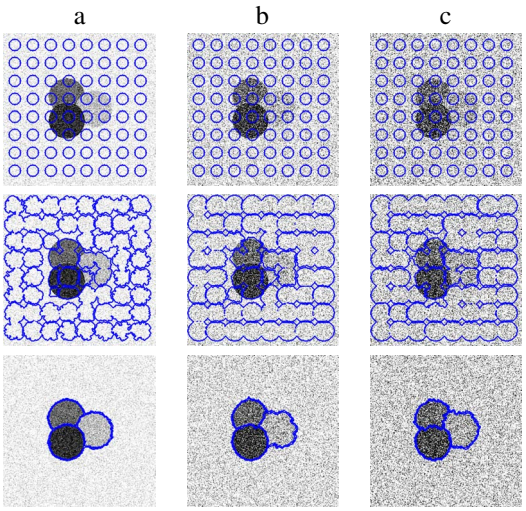


Figure 3. Segmenting the 'Icecream' image in Gaussian noise with variance 0.3 (column a), 0.5 (column b) and 1.0 (column c). Parameters were local window scale=3, λ=50, Gaussian model.

This segmentation region contains a mixture of the pixels from each of the adjacent image regions, and hence initially grows since adjacent pixels from both image regions match the local PDF. However, some time later adjacent segmentation regions will compete with it for ownership of its pixels and will win because it is cheaper to code those pixels as part of a homogeneous segmentation region than a hybrid region.

In the second experiment, another standard test image is used, 'Icecream' — three circles filled with different grey-values. The results are shown in Figure 3. The aim here is to demonstrate the robustness of the method to increasing amounts of noise. In this case, Gaussian noise has been used. Three cases are shown with increasing amounts of noise, each one is segmented using the Gaussian model and with the same parameters — local window scale=3, $\lambda = 50$. The segmentation is initialised with a regular grid of circular seed points.

The aim of the third experiment is to compare the relative performances of the Gaussian and the non-parametric region models. The image is the same as in the previous experiment, however this time with 'Salt and Pepper' noise added (density 0.5), causing the regions to be quite non-Gaussian. Figure 4 shows the results. Three parameters have been used for each region model; for the Gaussian case $\lambda = 15, 20, 25$ and for the non-parametric case (32 bin histogram) $\lambda = 50, 100, 200$. The local scale was set to 3 for all cases and the same initialisation from the previous

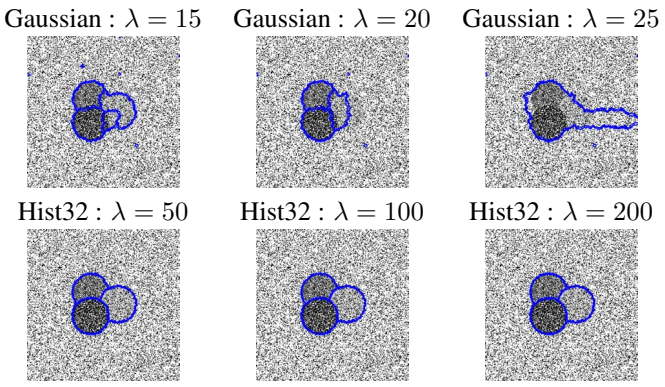


Figure 4. Segmenting the 'Icecream' image in 'Salt and Pepper' noise with density of 0.5, using Gaussian and non-parametric (Hist32) region models. Local window scale = 3

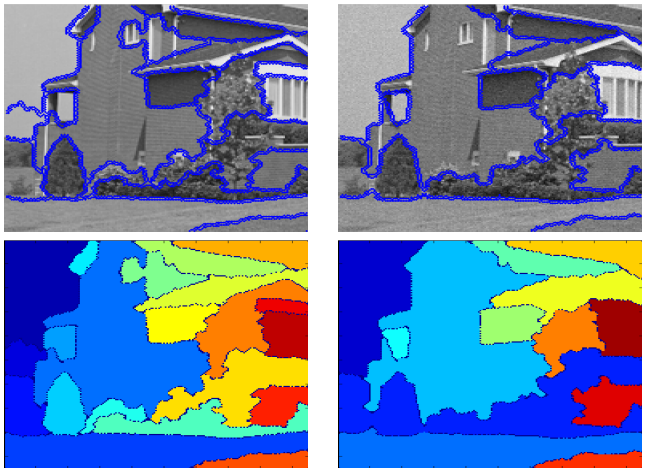
experiment was used.

The difference is quite clear. For the Gaussian case, only the $\lambda = 15$ case comes close to correctly segmenting the image, whereas in the non-parametric model case the correct segmentation is found in all cases, even though the λ varies over a larger range. We would expect that for non-synthetic images the non-parametric region model will make the algorithm more robust and less sensitive to precise setting of the parameters. Furthermore, a non-parametric region model becomes a necessity for segmenting textured images. The statistics of many texture descriptors, such as those based on wavelets, are typically non-Gaussian.

Typical execution times for these synthetic images (all 256x256 8-bit grey-levels) were in the range of 10-20 s[3].

In the fourth experiment, we have used a commonly used segmentation test image, 'smhouse', to ease comparison to results of previously published methods, such as [12]. The algorithm was run using the non-parametric region model (16 bin histogram) with parameters $\lambda = 100$ and local scale = 5. The initialisation was a regular grid similar to that used in the previous experiment. The results are shown in Figure 5(a). The segmentation is quite reasonable except where the sky has been partitioned into two regions. This is a problem with the region model. Though at first glance the sky region looks constant, it is in fact a gradient, ranging from grey-level 188 at the top to 174 at the horizon. Replacing the piecewise-constant model with a piecewise-smooth one would correct this problem. An alternative solution could be to increase the region cost λ, in effect encouraging regions to merge. However, in experiments we have found that contrary to expectations, other, more perceptually distinct re-

[3] All execution times are given for a AMD Athlon MP 2000+ machine with 1.5GB memory.



a b

Figure 5. Segmentation of original (a) 'smhouse' image and noisy (b) using non-parametric (Hist16) region model. Parameters λ=100, local window scale=5. The addition of the noise improves the segmentation.

gions, such as the bushes and lawn, merge at lower increments of λ than the two sky regions. There are several possible causes of this curious effect. One probable explanation is that the Region Competition cost function assumes a spatially constant noise model (Leclerc [6] derived MDL models for spatially varying noise). When this is not the case, regions with large variances are more likely to merge than those with small variances. A similar effect occurs in the non-parametric case; high entropy regions are more likely to merge than low entropy ones. In the 'smhouse' image, the sky region has a very tight variance whereas the bushes by the house exhibit a considerably larger variance. Recent graph-cut approaches can overcome this problem since they do not employ such strong region models [4, 16].

To illustrate this effect Figure 5(b) shows the results of the algorithm applied to the 'smhouse' image with added noise, using an identical set of parameters and initialisation as in the previous case. The addition of noise increases the variance (and entropy) of the sky region such that it can be grouped together. However, improved results aside, the concept of adding noise to an image to improve segmentation is somewhat counter-intuitive.

Another problem with our method can be seen in the region map image. Where there are a number of adjacent foreground regions, for example the bushes and the wall of the house, there is a small gap consisting of very small background regions. They are marked black on the region map. These occur because our speed function is binary; we set the speed to either +1 or -1. In most cases this does not cause a problem, but in some cases, such as these, it causes the competing foreground regions never to settle next to each other. A smoother speed function, one related to the statistical region forces, is necessary. Typical execution times for this image (190x190) were in the range of a 1-2 minutes for the non-parametric case (Hist16).

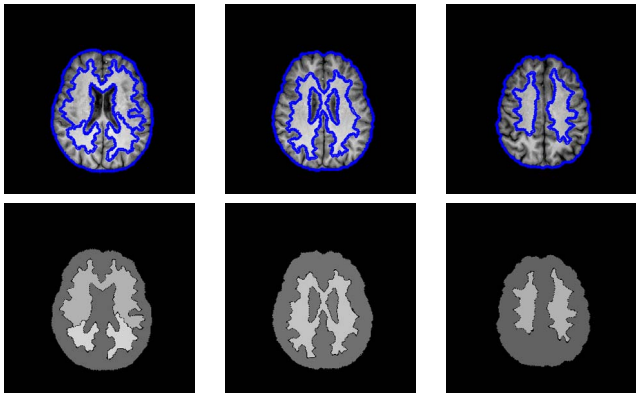


Figure 6. Segmentation of 'MyBrain' MRI brain scan images using non-parametric (Hist32) model. Parameters λ=200, local window scale=15, 5.

In the final experiment, the objective is to demonstrate that good results can be obtained over a class of images, in this case MRI images of the brain, using one set of parameters, $\lambda = 200$, local window scale=15 and 5. The two scales are used in a coarse-to-fine strategy: first, the segmentation is carried out at the coarse scale, and after convergence the fine scale is used. This is a common technique used to speed up convergence and allow smoother segmentations. The non-parametric model is used with 32 bins. A similar initialisation to previous experiments was used.

Three slices of the 'MyBrain' set are shown in Figure 6. Overall, the results are good. However, as with the 'smhouse' image, a smooth gradient in the central regions in the top slice has resulted in two regions being detected. MRI images are ideally piecewise constant up to the partial voxel effect in which tissues of two different types e.g. grey matter and white matter are present in the same voxel and the signal is a linear combination of the white and grey responses. However, imperfections in the B1 RF field generate a low frequency distortion known as the bias field distortion. The bias field can be estimated accurately for brain images and retrospectively corrected for [5, 18, 19]. A piecewise smooth region model could also be used to overcome this problem. Typical execution times for this image (256x256) were in the range of a 3-5 minutes for the non-parametric case (Hist32).

6. Summary and Conclusions

In this paper we have described an unsupervised segmentation algorithm based on Region Competition but implemented within a Level Sets framework. The main contributions are as follows:

Our Level Set algorithm can represent multiple classes using one embedded surface or phase. Previous methods such as [12, 13] require N coupled phases (or at best log N [3]) for N classes. This is beneficial for two reasons: firstly, it promises to be computationally more efficient; secondly it allows for dynamic splitting and merging of regions, hence facilitating unsupervised algorithms. It should be noted however that the execution times reported here are generally larger than those of previous methods such as [3, 12]. It should be noted that our implementation has not been optimised and there are likely to be many opportunities for computational savings. For example, our Narrow Band implementation uses a 'brute-force' approach to re-initialise the fronts whereas Fast Marching algorithms are a much faster alternative. In some applications memory requirement is also an important factor, for example 3D segmentation, and here the our approach should prove advantageous since it requires only one surface to be stored between iterations.

Formulating the Region Competition algorithm in a Level Set framework overcomes the implementation difficulties associated with the pixel list approach, since the Level Set technique can implicitly handle topological changes such as merging and splitting and is essentially a curve evolution technique. Finally, we have generalised the standard Region Competition expressions for competitive and merging stages. This generalisation makes the algorithm more robust and less dependent on a careful choice of parameters. We have provided results of our algorithm applied to a number of synthetic and real images.

Acknowledgments

This work was sponsored by a Motorola University Partners in Research grant. We would like to thank Dr. Paola Hobson of the Motorola UK Research Laboratory for the many discussions we have had during the course of this work. Image set 'MyBrain' from FMRIB, Oxford University.

References

[1] D. Adalsteinsson and J. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, 1995.

[2] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. In *Proc. International Conference of Computer Vision*, pages 694–699, 1995.

[3] T. Chan and L. Vese. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision*, 50(3):271–293, 2002.

[4] P. Felzenszwalb and D. Huttenlocher. Image segmentation using local variation. In *IEEE Computer Vision and Pattern Recognition*, pages 98–104, 1998.

[5] R. Guillemaud and J. Brady. Estimating the bias field of MR images. *IEEE Trans. on Medical Images*, 16(3):238–251, 1997.

[6] Y. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision.*, 3(1):73–102, 1989.

[7] F. Li, Z. Xie, and J. Brady. Texture segmentation using local energy in wavelet scale space. In *Proc. European Conference of Computer Vision*, pages 304–313, 1996.

[8] S. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, Tokyo, 1995.

[9] D. Marr. *Vision: a Computational Investigation into the Human Representation ond Processing of Visual Information*. W. H. Freeman, San Francisco, 1982.

[10] D. Mumford and J. Shah. Boundary detection by minimizing functionals. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 22–26, 1985.

[11] S. Osher and J. Sethian. Fronts propagating with curvature speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[12] N. Paragios and R. Deriche. Coupled geodesic active regions for image segmentation: A level set approach. In *Proc. European Conference of Computer Vision*, 2000.

[13] N. Paragios and R. Deriche. Geodesic active regions and level set methods for supervised texture segmentation. *International Journal of Computer Vision*, 46(3):223, 2002.

[14] J. Rissanen. *Minimum Description Length Principle*, pages 523–527. Wiley, New York, 1985.

[15] J. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Science*. Cambridge University Press, 1999.

[16] J. Shi and J. Malik. Normalized cuts and image segmentation. In *CVPR*, pages 731–743, 1997.

[17] Z. Tu and S.-C. Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(5):657–673, 2002.

[18] W. M. Wells, E. L. Grimson., R. Kikinis, and F. A. Jolesz. Adaptive segmentation of MRI data. *IEEE Trans. on Medical Images*, 15(4):429–442, 1996.

[19] Y. Zhang, J. Brady, and S. Smith. Segmentation of brain MR images through a hidden Markov Random Field model and the expectation maximization algorithm. *IEEE Trans. on Medical Images*, 20(1):45–57, 2001.

[20] H.-K. Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *Journal of Computational Physics*, 127:179–195, 1996.

[21] S.-C. Zhu and A. Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(9):884–900, 1996.