Applications of the "Creep–and–Merge" System: Corner Detection

Antranig Basman, Joan Lasenby and Roberto Cipolla
Department of Engineering,
University of Cambridge
Cambridge CB2 1PZ, UK
[amb26|jl|cipolla]@eng.cam.ac.uk

Abstract

The Creep–and–Merge (CAM) segmentation system (described in [3, 2]) is a novel architecture for region-based segmentation; it is designed to be efficient, insensitive to noise, scale and image geometry, capable of applying the widest range of statistical models, and to contain no adjustable parameters. This paper describes the continuing development of higher–level processing models utilising this system: we present a geometric ("true") corner detector with good performance, and give qualitative and quantitative comparisons with other leading systems.

1 Introduction

The most popular feature extractors for computer vision tasks are still edge and corner detectors, based on local image measurements from a fixed–sized neighbourhood. Competing region–based schemes making use of more powerful global methods and constraints currently suffer from poor computational performance and generality, and often fail to deliver the level of robustness and automation which this alternative technology promises.

Two region–based segmentation paradigms (the "snakes/balloons" framework put forward in [7, 4], and generalised in [13], and split–and–merge/link, presented in [8, 1]) and generalised in [11]) can be used to classify the majority of region–based approaches, but these suffer from various problems. The former are still predominantly local methods, and also suffer from troublesome discretisation of curvature and gradient terms, and topological difficulties (curve self-intersection, etc.); whereas algorithms in the latter class are generally fragile. Since they progress in a fixed number of recursive passes through the image, it is easy for them to overlook important minima in their energy functions.

More recent developments ameliorate these problems, but [11] still has problems with regularisation of the number of regions, and resolution of extracted boundaries, as does [13], even though it belongs to the second camp.

The CAM system may either be viewed as an explicit discretisation of a contour–based scheme, or as an iterative enhancement to region splitting methods — it shares in the good properties of both.

BMVC 1998 doi:10.5244/C.12.59

2 Framework and Assumptions

2.1 Configuration

We model the image as a completely covered by non-intersecting connected regions $\{R_i\}$. Each region contains data D_i with $N_i = N_i(D_i)$ samples, drawn from $\boldsymbol{\alpha}(D_i)$, one of a collection of models indexed by parameter vector $\boldsymbol{\alpha}$.

The configuration is optimised entirely in terms of quantities known as *discriminabilities* $\delta(D_i, D_j)$ of two sets of region contents, which are the only communication between the geometric and statistical parts of the system. This function represents, in some suitable metric (mapping to real numbers), our view of the likelihood that the adjacent regions truly are drawn from different models. In the simplest framework, this would be just a function of grey-level differences.

A number of choices are suitable for this function, depending not least on the statistical models chosen, but also the statistical paradigm adopted; however in general δ is to have the dimensions of a log probability.

2.2 Discriminabilities

Let H_ω be the *null hypothesis*, that two regions are described by the same model, and H_a be the *alternative hypothesis*, that they are described by different models.

In the *frequentist* framework, these become the point hypotheses $H_\omega : \boldsymbol{\alpha}(D_{ij}) = \hat{\boldsymbol{\alpha}}_{ij}$ and $H_a : \boldsymbol{\alpha}(D_i) = \hat{\boldsymbol{\alpha}}_i$ and $\boldsymbol{\alpha}(D_j) = \hat{\boldsymbol{\alpha}}_j$, where the hat symbol represents the maximum likelihood estimate of the relevant parameter, and D_{ij} is the data formed by merging the two regions; we apply the *Likelihood Ratio Test*, and a suitable choice for δ is

> The log level of confidence at which the null hypothesis H_ω that D_i and D_j are drawn from the same distribution would be rejected.

In the *Bayesian* framework, we are required to integrate over all possible values for the parameters $\boldsymbol{\alpha}$ under the respective hypotheses, and δ becomes

> The log *Bayes factor* (ratio of evidences) in favour of the hypothesis that D_i and D_j are from the same distribution, against the hypothesis that they are drawn from different distributions.

For illustrative purposes, we show the two possibilities for δ in the case where our models are univariate Gaussians $G(x; \mu_i, \sigma_i)$, and our parameters are the means μ_i and standard deviations σ_i, and our models allow differences in both parameters. In this case, both functions factor into three terms $K(D)$, so that $\delta(D_i, D_j) = K(D_{ij}) - K(D_i) - K(D_j)$. This does not happen in general. Table 1 shows the values for $K(D)$; s is the maximum likelihood estimate for σ.

Framework	Discriminability Function
Frequentist	$N \log(s^2)$
Bayesian	$\log\big(\Gamma(N/2)\big) - (N/2)\log(s^2) - \log(N/2)$

Figure 1: Terms in the discriminability functions under different paradigms for univariate Gaussian distributions

2.3 Comments

At this point, we note only that the Bayesian function involves significantly more computation, and that although it performs fractionally better in our application than the frequentist measure (as it must, given the proven optimality of the Bayesian framework in general), this improvement is not justified by the increase in computation. We must bear in mind that this function will be evaluated many millions of times in segmenting even modest sized images.

Both quantities behave similarly – borders with lower (more negative) discriminabilities are said to be *strong* borders; these values can be much below -1000 for some artificial images. Borders with higher (less negative) discriminabilites (close to 0) are *weak* borders. Bayesian discriminabilites can rise marginally above 0.

3 Optimisation

3.1 Operations

During optimisation, the configuration is perturbed repeatedly, by means of various operations; these comprise

- Splitting (Smashing) — a region is subdivided into smaller regions.
- Creeping — the boundary between two regions is deformed.
- Merging — two adjacent regions are combined.

Some algorithms lack one or more of these operations, and suffer often in efficiency, and usually in performance — if the operation set is impoverished, it is very easy for a system to get trapped in undesirable configurations.

There is not space here to describe the full system (see [2]); in outline, splitting and merging occur when the discriminability δ is on the wrong side of a confidence threshold t_c. This parameter represents the statistical certainty we require in order to maintain two regions as distinct. A value of around -5, representing a confidence of about 98% in decision-making, is appropriate, and does not need external adjustment.

3.2 Scale Uncertainty

The most important feature of the CAM algorithm is that it dynamically adjusts *all* of its internal parameters in response to the image data. Many algorithms give spurious or incorrect outputs when presented with image data that is only mildly corrupted by noise, and few exhibit useful behaviour as noise becomes more severe. What [12] term the *Uncertainty Principle* in image processing is the solution — the greater the difficulty of the statistical decision task, the coarser the scale at which it can be reliably made.

It is therefore important that all region boundaries are only represented at an appropriate scale for the discrimination that gave rise to them. In the CAM algorithm, we define, using the discriminability $\delta(D_i, D_j)$ of two neighbouring regions, a *magic number* N^*, which represents, in pixels, the minimum area by which the boundary can be perturbed during Creeping to produce a statistically distinguishable configuration.

To define N^*, we consider hypothetical subregions R_i^N of R_i with size N, which we assume to give rise to the same maximum likelihood model vector $\hat{\boldsymbol{\alpha}}_i$. We define

$$N^*(D_i, D_j) = \max N \quad \text{such that} \quad \delta(D_i^N, D_j) < t_c \tag{1}$$

Since we use a quadtree framework in our current implementation, this indicates that the boundary of the regions R_i and R_j is kept sampled to a spatial resolution of close to $\sqrt{N^*}$ pixels, using square cells.

With these definitions of N^* and t_c, the number of regions and the resolution of each boundary in the system are continuously kept correctly adjusted, without the need for the user to set scale or "busyness" parameters before work begins.

4 Performance

We present some quantitative results on the performance of the raw CAM system before passing on to the higher-level systems that have been constructed on it.

[6] contains a description of a suite of test images and evaluation criteria that can be applied to segmentation systems, together with a collection of test results for five competing region– and edge–based systems. We apply the same tests to this algorithm — the images segmented are similar to that in Figure 3, with differing signal–to–noise ratios (SNR). Our definition of SNR is also taken from [6], since their definition is not the usual one.

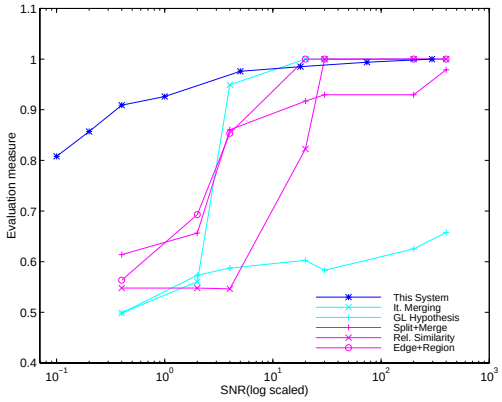


Figure 2: Performance evaluation

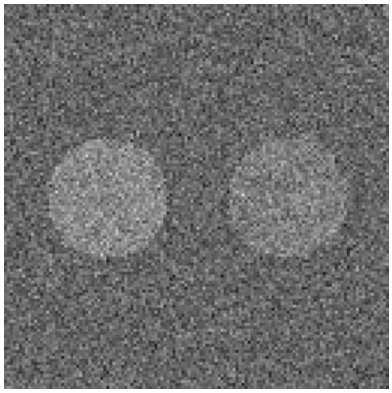


Figure 3: Typical image from [6]

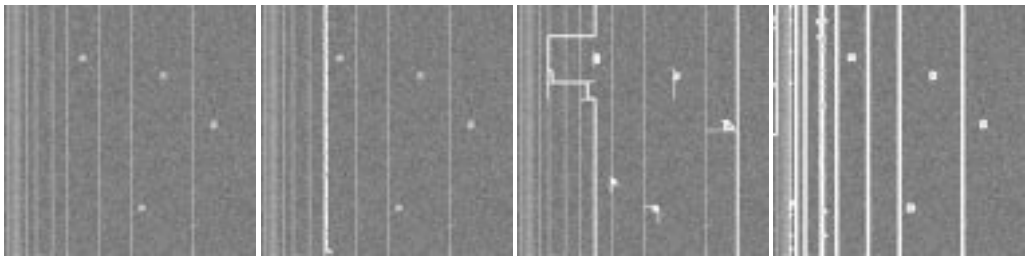
Figure 2 shows the considerable superiority of our method in noisy images, continuing to give good results in increasing noise long after the other systems have given up. The system also performs very well in low–noise images – since it has no smoothness regularisation, it misclassifies a few pixels in moderate noise. This could be remedied by even a simple higher–level geometric system.

It should also be noted that the results presented for the other five systems are the result of optimising the criterion with respect to their adjustable parameters, over several dozen runs. Our system also scores heavily over the others in that it has no such parameters.

In the other direction, Figure 4(d) shows the successful segmentation of Figure 4(a), taken from [13]; the current implementation correctly identifies the thin lines and blocks

until quite narrow spacings at the left; the left–hand strip is seen to be a single, high–variance region. The noisy boundary of the fifth bar from the right is due to sampling problems in the quadtree. Enhancements are planned to enable the system to segment this image completely correctly.

It should be noted that contrary to the claim made in [13], this success has been made in a unified framework without the addition of any form of edge term.



(a) (b) (c) (d)

Figure 4: (a) Original degenerate image; (b) Result from Yuille+Zhu system[13]; (c) Split and merge result; (d) CAM system.

5 Further Processing — Medical Images

Clearly, real images are not described by the piecewise Gaussian models of the previous section — in order to address this issue, we may be tempted to adjust the confidence parameter t_c downwards, in order to provide more "sloppy" segmentations, and hope to "segment out" variations in the images we are not interested in; this is a common approach.

There are two main problems with this idea; i) we have introduced an unsettable, image–dependent parameter into the system, and ii) the use of an unnatural probability threshold causes systematic bias in the resulting contour positions.

Therefore, all higher–level processing systems will make use of the CAM algorithm in its natural, parameter–free form.

In this form, the system will clearly produce very many more regions than will be desirable for most applications; it will "over–segment". However, we can observe that the algorithm has (hopefully) detected *all* statistically significant boundaries within the image, so the contours we are interested in, if they are visible at all, must be accurately represented as a subset of the produced contours. Our problem is thus one of selection, a much easier task than the original segmentation task.

As it transpires, the problem is so simple that an extremely crude higher level will do a surprising amount of the work required to segment quite difficult images — we use the "roundness" metric, which is the squared perimeter divided by the area of the region formed by aggregating sub-regions produced by raw CAM.

Figure 5 shows the original ultrasound image, containing a considerable amount of noise, and a fairly irregular contour. Figure 6 shows the raw CAM segmentation — it can be seen that all parts of the required contour are represented in the "jigsaw" middle image.

Starting from a central region that is darker than all of its neighbours, a simple optimised expands the contour by aggregating more of the small CAM regions, optimising

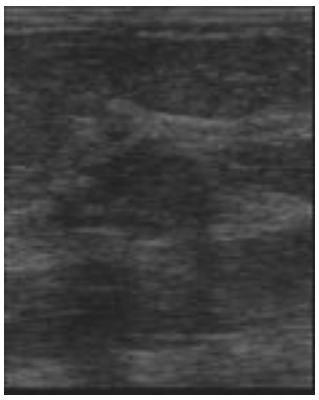


Figure 5:
Original ultrasound image

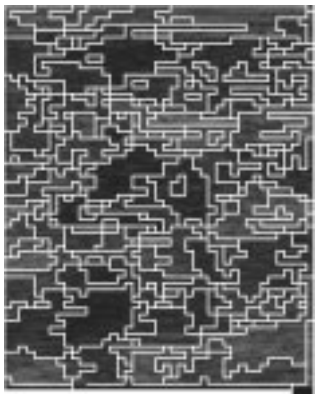


Figure 6: Raw segmentation

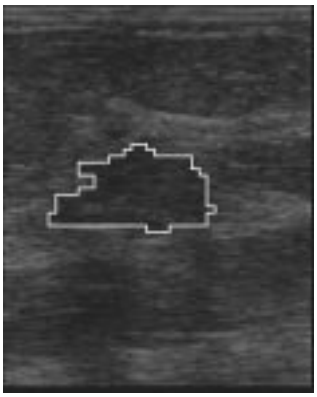


Figure 7:
Aggregated result

the roundness criterion. The final result is shown in Figure 7. More details are found in [2], and an interactive system based on this idea is described in [9]. Work is underway on a more sophisticated higher–level system, based on the results in the next section, for the segmentation of the most difficult images.

6 Corner Detection

6.1 Statistics

We will model points $\{x_i, y_i\}$ on the contour as arising with probability dependent on a Gaussian function of their perpendicular distance from a straight line. Thus our models have 3 parameters, a, b and c, the location parameters of the straight line segment in question, described by the equation $ax + by + c = 0$; we will encode σ, the standard deviation of the Gaussian, by $1/\sqrt{a^2 + b^2}$. In this case, we will not segment on the parameter σ.

Again, we have two choices of statistical test to apply, the Frequentist, and the Bayesian. In this case, the behaviour of the two is more clearly differentiated — the Bayesian test is given in terms of the Gaussian Hypergeometric function $F(-N, -N; 1, x)$ where N is the number of data points, and x is a function of their angular spread. The derivation of this is outside the scope of this paper — however, the series expansion for this function converges extremely slowly, making this test almost two orders of magnitude slower than the Frequentist form.

The Frequentist criterion (equivalent of $K(D)$ in Figure 1) is $-(N/2)\log(\lambda)$, where λ is the minimum eigenvector of the matrix $\mathrm{Covar}(x_i, y_i)$. This can be easily evaluated, and interestingly proved more sensitive to corners in low-data situations.

6.2 Algorithm

The generality of the CAM framework becomes apparent, since we will apply the 1–dimensional version of the *same algorithm* on the extracted contours to detect corners; we will segment them into piecewise linear sections.

In this case, hierarchical subdivision has no performance benefits, since the regions are all just intervals - therefore splitting can occur at arbitrary points. The other stages of the algorithm are unchanged.

6.3 Results

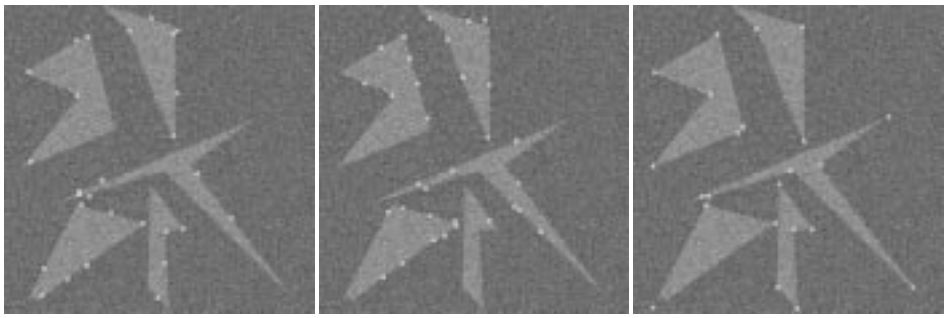


Figure 8: Harris detector Figure 9: SUSAN detector Figure 10: CAM detector

Figures 8, 9 and 10 show an automatically generated image containing five polygons, amid a moderate amount of noise (signal 32 intensity units difference, noise standard deviation 16). From left to right are shown the results of the Harris corner detector ([5]), the SUSAN corner detector ([10]), and the CAM system.

Note that the first two systems have more than one adjustable parameter, and the results shown were the best obtainable by sweeping these over their ranges — the Harris corner was explictly told to look for less than 30 corners, and with the SUSAN detector we hunted around until a threshold value of $t = 30$ gave better results (this is an inverse scale — lower values of t produce more corners).

The CAM corner system may allow the adjustment of the parameter t_c for the *second*, 1–dimensional segmentation, but a value of -50 has been found sufficient for all images so far examined. This much lower number reflects the many other sources of error factored into the contour generation, such as statistical variations in the original image, segmentation errors, and discretisation errors. These indicate we require greater evidence in order to reliably detect model changes (corners).

7 Performance Evaluation

System	True Detects	Total Detects	Detect Quality	RMS pos. error
Harris	16	17	64%	1.2
SUSAN	19	21	73.1%	0.48
CAM	22	23	88%	0.47

Figure 11: Corner detection performace for SNR = 8

Figure 11 clearly shows superior performance of CAM even in light noise; the image used was similar to that of the figures above, but the noise was only of standard deviation 4 units, giving a SNR of 8. The image contains a total of 24 true corners — of these, one

at the top of the scene is never detected by any scheme, since it lies exactly on the upper edge of the image. We ascribe a "Detection Quality Rating" as the ratio of

$$\frac{\text{True Detects}}{\text{Total Actual Corners} + \text{False Detects}} \tag{2}$$

We normalise the distance to the true position of the corner used as the basis of the RMS calculation as follows: the "narrow angle", θ, of the corner, is defined as either the angle in the corner, or its complement, depending on which of them is acute. We then scale the component of the error vector parallel to the bisector of the "narrow" angle by $\sin(\theta)$, and leave the perpendicular component unchanged. This is to prevent unavoidable larger positional errors in narrow corners from swamping the total RMS measure.

7.1 Parameter Robustness

We return to the issue of parameter robustness — again, Figure 11 was generated by searching over the parameter spaces of Harris and SUSAN for optimal results, while CAM was run at its standard settings. The graph in Figure 12 examines in detail exactly how stable the various algorithms are — we also allow the t_c parameter of CAM to be adjusted, to see the change in system performance.

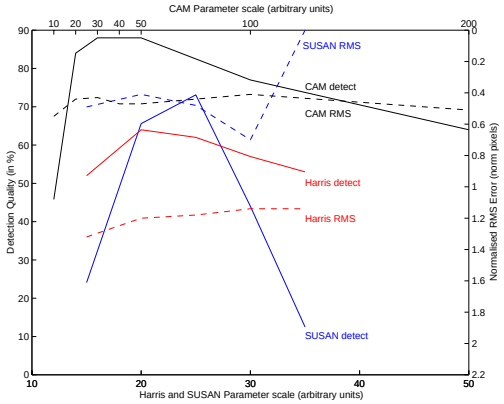


Figure 12: Parameter robustness for SNR = 8

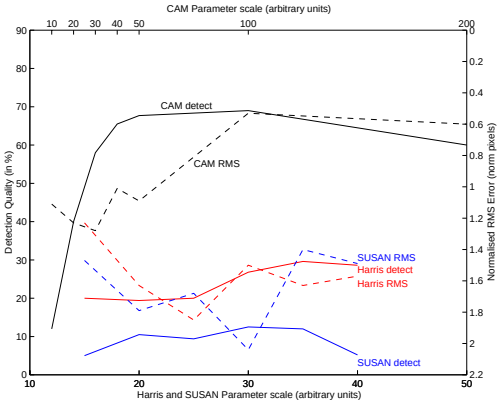


Figure 13: Parameter robustness for SNR = 2

This graph requires some explanation: the most important point is that the horizontal scale has no common meaning for the different algorithms, since their individual thresholds are in some arbitrary scale of units — the three systems have been plotted on one graph purely for convenience. The dashed lines represent the RMS performance for each algorithm, which should be referred to the right–hand axis; the solid lines represent the detect performance, which for which the scale appears on the left axis. Finally, the top of the graph represents better performance; lower RMS error, and better detect rating.

What matters is not the horizontal scale of the graphs, but their shape — a peaked or curved shape indicates that the algorithm's performance depends sensitively on the parameter setting being varied, whereas a flattened shape indicates that the algorithm is robust. Note that as well as being flat over a wide range, the line representing CAM detect performance is significantly higher than that of the other systems.

Regarding location performance, the Harris corner detector, as many studies report, is here shown to be inferior in location performance, but SUSAN and CAM are equivalent in this regard.

Note that although further processing can enable the SUSAN edge detector to give subpixel output, the same does not seem to be true of the SUSAN corner detector. Since for this test image, the true corner coordinates were in fact integers, a part of SUSAN's good location performance was due to getting a number of the corners precisely correct. Further tests with subpixel test images are expected to show this performance to degrade.

Figure 13 plots the same quantities for SNR $= 2$, the same as that in the above figures. The SUSAN performance now almost negligible, and the Harris performance has halved. But note also that as well as still being peaked, the position of maximum performance of these two algorithms has *changed* in threshold space — the threshold parameters for these algorithms are truly unsettable, since we cannot determine their optimum value without first looking at the image. As well as being flat, the value -50 was still well within the flat region for CAM.

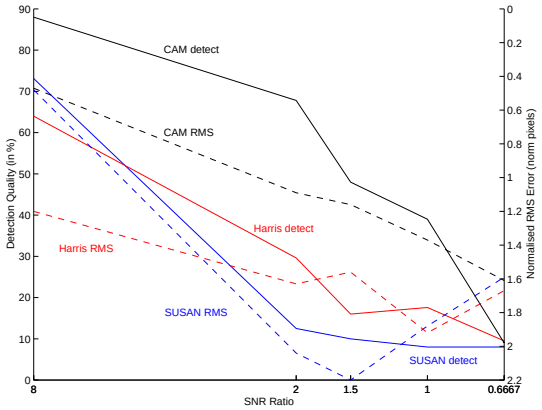


Figure 14: Breakdown of detection with decreasing SNR

7.2 Noise Breakdown

Finally, we show in Figure 14 the breakdown of the algorithms under increasing noise — again, we use the same settings for CAM, and pick optimum ones for Harris and SUSAN; by the time we reach the right–hand point of the graph, the optimum threshold value for t had drifted up to 50. Note that for very low–noise images, the authors recommend a value of $t = 10$. l Again, the CAM algorithm wins out uniformly in detect performance.

8 Notes on Performance

In addition to its function as a corner detector, the CAM algorithm, since it globally segments the image, has other benefits too; we may output a richer set of descriptors (angles, contrasts, distances, etc.) that will aid a yet further stage of processing, for example feature matching, or object recognition. Being statistically general, CAM is also not limited to finding linear joints; future versions of the system are planned to discover ellipse segments, and finally spline segments.

However, this performance currently comes at a computational price; while the highly optimised C implementation of SUSAN can process an entire image in 0.3 seconds, the CAM system is more than an order of magnitude slower, taking around 15-20 seconds to process an image with the (largely unoptimised) C++ backend, and a further 1-2 seconds to detect corners with the Java frontend.

The system is still under development, but it is considered unlikely that the total processing time could be brought to much under 5 seconds on today's hardware. Thus, for real–time applications, SUSAN remains the system of choice; however, for the future, and for today's offline reconstruction and matching applications, the CAM system promises many benefits.

References

[1] H.J. Autonisse. Image segmentation in pyramids. *Computer Vision, Graphics and Image Processing*, 19(4):367–383, 1982.

[2] A.M. Basman, J. Lasenby, and R. Cipolla. The Creep–and–Merge segmentation system. Technical Report CUED/F-INFENG/TR295, University of Cambridge, July 1997.

[3] A.M. Basman, J. Lasenby, and R. Cipolla. Efficient region segmentation through 'Creep–and–Merge'. In *Proc. 9th Int. Conf. on Image Analysis and Processing*, volume I, pages 223–230. Springer-Verlag, 1997. LNCS 1310.

[4] L.D. Cohen. On active contour models and balloons. *Computer Vision, Graphics and Image Processing*, 53(2):211–218, May 1991.

[5] C.G. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, 1988.

[6] H. Jiang, J. Toriwaki, and H. Suzuki. Comparative performance evaluation of segmentation methods based on region growing and division. *Systems and Computers in Japan*, 24(13):28–42, 1993.

[7] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proc. 1st Int. Conf. on Computer Vision*, pages 259–268, 1987.

[8] T. Pavlidis. *Structural Pattern Recognition*, chapter 5. Springer-Verlag, 1977.

[9] C. Sherratt, A. Basman, R. Prager, and A. Gee. Interactive segmentation of ultrasound images by Creep–and–Merge. In *Medical Image Understanding and Analysis*, pages 133–136. Leeds, 1998.

[10] S.M. Smith and J.M. Brady. SUSAN — a new approach to low level image processing. *Int. Journal of Computer Vision*, 23(1):45–78, May 1997.

[11] M. Spann and A.E. Grace. Adaptive segmentation of noisy and textured images. *PR*, 27(12):1717–1733, December 1994.

[12] R. Wilson and G. Granlund. The Uncertainty Principle in image processing. *IEEE Trans. Pattern Analysis and Machine Intell.*, 6(6), November 1984.

[13] S.C. Zhu and A. Yuille. 'Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multiband Image Segmentation'. *IEEE Trans. Pattern Analysis and Machine Intell.*, 18(9):884–900, September 1996.