



# GPU-accelerated image segmentation based on level sets and multiple texture features

Daniel Reska<sup>1</sup> · Marek Kretowski<sup>1</sup>

Received: 20 April 2020 / Revised: 9 September 2020 / Accepted: 16 September 2020 /  
Published online: 3 October 2020  
© The Author(s) 2020

## Abstract

In this paper, we present a fast multi-stage image segmentation method that incorporates texture analysis into a level set-based active contour framework. This approach allows integrating multiple feature extraction methods and is not tied to any specific texture descriptors. Prior knowledge of the image patterns is also not required. The method starts with an initial feature extraction and selection, then performs a fast level set-based evolution process and ends with a final refinement stage that integrates a region-based model. The presented implementation employs a set of features based on Grey Level Co-occurrence Matrices, Gabor filters and structure tensors. The high performance of feature extraction and contour evolution stages is achieved with GPU acceleration. The method is validated on synthetic and natural images and confronted with results of the most similar among the accessible algorithms.

**Keywords** Image segmentation · Active contour model · Level set method · Texture analysis · GPU acceleration.

## 1 Introduction

Image segmentation is one of the most fundamental problems in computer vision. Deformable models [34] are a successful class of segmentation algorithms based on the idea of a deforming shape that adapts to the desired image region. The fundamental form of the deformable model-based segmentation method was proposed by Kass et al. [25] as an active contour model (ACM), also known as a “snake”. The snake model is a parametric curve with an evolution process controlled by a set of external and internal energies. External energies attract the shape to the desired image area and move it towards the boundaries of the segmented region, while the internal forces control the contour smoothness. This method

---

✉ Daniel Reska  
d.reska@pb.edu.pl

Marek Kretowski  
m.kretowski@pb.edu.pl

<sup>1</sup> Faculty of Computer Science, Białystok University of Technology, Wiejska 45A,  
15-351 Białystok, Poland

overcomes many problems, like image noise and boundary irregularities, and makes it easy to extend with new types of image features and constraints. These advantages made the classical ACM very influential and widely improved, e.g. with the introduction of expansion forces [10], edge-based vector field energies [58], and region-based image energies [44].

The original ACM also had some drawbacks, like difficulties in topological adaptability (requiring additional mechanisms [30]) and sensitivity to initialisation. A major improvement came with the incorporation of the level set framework, proposed by Osher and Sethian [37, 47]. Instead of the explicitly defined parametric contour, the level set approach proposed an evolving surface, where the contour is implicitly represented as a set of surface points that have their height equal to zero (the zero level set). During the surface evolution, the zero level set can naturally split or merge and can disappear or appear in locations not constrained by the contour initial form. Notable level-set based models include Geometric [6] or Geodesic Active Contours (GAC) [7] and region-based Active Contours Without Edges (ACWE) [9].

Traditionally, the image energies of popular ACMs relied either on edge information [6, 7, 25, 58] or on intensity statistics of the segmented regions [9, 44], making the models best suited for regions with distinctive borders and fairly uniform intensity. These approaches, however, are often not sufficient for segmentation of non-uniform textured regions with high contrast and/or directional patterns. Although the discrimination of texture patterns is easy for the human eye, in computer vision it is still hard to find a universal texture descriptor. Due to the lack of a unified texture model, the segmentation methods rely on various feature extraction methods [40], like Grey Level Co-occurrence Matrices (GLCM) [20, 49], Gabor filters [24], structure tensors [4, 5] or histogram analysis [28, 36].

Texture segmentation using active contour models has been a popular topic of research in recent years [1, 14, 17, 51, 52]. Particularly, the variational ACWE-like models and their modified multi-channel versions for vector-valued images [8] have been often employed [1, 19, 32, 57]. Incorporation of the texture features into the ACM-based method usually encounters significant difficulties. First of all, the method must settle on a set of many possible feature extraction approaches and typically, only one approach (like for example Gabor filters) is adapted. Obviously, it strongly restricts the robustness and generality of such methods. Furthermore, even with this simplification (i.e. selection of a single texture feature extraction approach), the algorithm should be able to deal with multiple feature channels. As the level set-based methods can be computationally intensive, it can have a significant impact on the performance. As a result, these multiple channels are often combined/aggregated to avoid overcomplicating the model. It is why new solutions are required that are ready to integrate various texture characteristics and smartly process many features.

In this paper, we present a novel level set-based method for image segmentation that addresses the mentioned problems of the texture analysis integration into the active contour model. The method is not tied to any particular texture descriptors and inherently is able to consider multiple feature types. One can dare to say that the more approaches are used, the better the results could be. Here, the proposed solution is demonstrated with just three texture feature extraction approaches, namely GLCM, Gabor and structure tensor. The segmentation algorithm is divided into three main stages. It starts with many features calculation and selection of the most adequate descriptors for the considered (manually initialised) region of interest. It is followed by a fast level set stage that results in a rough segmentation. Finally, an ACWE-based stage fine-tunes the contour and provides the final result. Implementation of the first two stages makes heavy use of Graphics Process Unit (GPU) acceleration and exhibits a reasonable performance even on mid-range graphics

cards. Even if the building blocks of the proposed algorithm are known, their joint use is undoubtedly original and open to further extensions.

The main objective of this work is a new fast segmentation method that is able to use a relatively large bank of different features and perform the costly level set evolution while maintaining computation time suitable for interactive usage. The method combines both classical and region-based level set schemes and is open to extension with additional descriptors.

The concept of a multi-stage approach was initially outlined in [42] in the context of classical discrete parametric contour (snake). Apart from the similarities in the feature selection stage, the snake-based model employed a vastly different evolution scheme that considered only the closest neighbourhood of the contour points. In contrast, the proposed approach uses both local and global level set-based ACMs that are more robust in handling of multiple features and offer little topological restrictions.

The rest of the paper is organised as follows. Section 2 gives a brief review of the up-to-date related work on texture segmentation. It also recalls the utilised feature extraction and level set methods. Section 3 describes the proposed method and discusses implementation concerns. Section 4 presents an experimental evaluation of the method on both synthetic and natural images, as well as a comparison with other segmentation methods. Conclusions and possible future works are given in the last section.

## 2 Background

### 2.1 Related work on texture segmentation

Active contour-based methods were widely adapted to the task of texture segmentation. While the parametric contours are still used [33, 52], the level set-based models are gaining more and more attention. Generally, texture-based methods can be divided into supervised and unsupervised. The supervised methods often take advantage of information from the initial texture analysis stage. Notably, Paragios and Deriche [38] proposed an extension of the GAC model that performs an off-line feature analysis. The method requires a set of texture patterns present in the image. The analysis results are then integrated in the region and boundary image forces. Pujol and Radeva [39] employed the GLCM features in their ACM. Linear Discriminant Analysis was used to reduce a feature space, which was then applied to create a likelihood map that guided the contour deformation.

In the case of the unsupervised approaches, no prior analysis or classification is performed. Shen et al. [48] proposed a method for creating a likelihood deformation map from the intensity statistics and edge information of the initial region. This solution was integrated with 2D and 3D models. Huang et al. [23] presented a similar approach, where the intensity characteristics were used for the textures with small texons, and a bank of Gabor filters was utilised in the case of textures with larger patterns. Awate et al. [3] proposed a fast multi-phase level-set method based on threshold dynamics [15] that introduces a general nonparametric statistical image neighbourhood model instead of specific texture features. Rousson et al. [45] presented a method based on a variational framework, where the texture features are extracted with a structure tensor and nonlinear diffusion scheme. Savig et al. [46] created a texture edge detector from the Gabor feature space based on the Beltrami framework [50] and integrated it with the GAC and ACWE models. Houhou et al. [22] also used the Beltrami framework to propose a descriptor that captures both edge and textural properties. In this case, the ACM was based on Kullback-Leibler distance.

More recently, Wu et al. [57] presented a method based on ACWE, where GLCM and Gabor features are fused together to create the final feature space. Min et al. [32] integrated the image intensity and a novel adaptive texture descriptor into the ACWE-based model. Mewada et al. [31] combined the ACWE approach with a modified version of the linear structure tensor. The ACWE model was also applied for texture segmentation by Wang et al. [55], where the contour was driven by a local histogram. Tatu and Bansal [53] proposed a GAC-based model that uses intensity covariance matrices for the texture features. Gao et al. introduced a factorisation-based ACM that utilises the local spectral histogram as the texture feature [16], and, more recently, a model that performs a fusion of intensity and Gabor-based features along with a factorisation scheme [17]. Dong et al. [14] also employed a factorisation-based ACM, combined with neutrosophic sets, in the task of color texture segmentation. Dahl and Dahl [11] created a method based on ACWE and probabilistic image patch dictionaries. Ahmad et al. [1] proposed a fuzzy version of a variational level set model that uses the coefficient of variation as a region statistics. The model was proven to be effective on textured and inhomogeneous images [59]. Wang et al. [56] also tackled this problem by introducing an inhomogeneity entropy descriptor that was driving the energy of a level set-based ACM.

## 2.2 Texture feature extraction

**Grey Level Co-occurrence Matrix** The GLCM is a square matrix of order equal to the number of pixel grey levels. For a spatial window  $w$  on an image  $I$ , the GLCM contains the probabilities of pair-wise occurrence of pixels with two grey levels. Each of the  $(i, j)$  matrix elements contains a number of occurrences, where a pixel with intensity  $i$  is adjacent to a pixel with intensity  $j$ . The adjacency of the pixels is defined by the distance between them in a given orientation. This matrix, after normalisation, is then used to calculate a set of texture feature descriptors, like Entropy, Correlation, Homogeneity, Contrast or Energy [20]. The feature generation process usually considers a selected set of features and generates a feature space with a combination of parameters: window size, pixel distance and orientation. To capture the properties of various patterns in different scales and orientations, the highly-dimensional space can contain a large number of features, therefore a feature selection [39] or fusion [57] is often performed.

**Gabor features** A two-dimensional Gabor filter can be defined as a Gaussian kernel function modulated by sinusoidal wave plane. In this form, the response of the filter can be obtained by its convolution with the input image. Gabor filters were designed to model the behaviour of mammalian visual cortex cells [12] and have the ability to localise the information in both spatial and frequency domain. These properties make them useful in the image feature extraction process, particularly in the texture analysis field [24, 40]. In practice, the feature extraction is often performed by convolving the image with a set (bank) of Gabor filters with different orientations, scales and wavelengths.

**Structure tensor** For a scalar two-dimensional image  $I$  with a pixel coordinate  $p = (x, y)$ , the structure tensor [4, 5] is defined as a matrix derived from the gradient of the image:

$$S(p) = \begin{pmatrix} g(p) * I(p)_x^2 & g(p) * I_x(p)I_y(p) \\ g(p) * I_x(p)I_y(p) & g(p) * I_y^2(p) \end{pmatrix} \quad (1)$$

where  $g(p)$  is a Gaussian kernel with a  $\sigma$  standard deviation and  $I_x$  and  $I_y$  are partial derivatives of  $I$  in a window  $w$  centred at  $p$ . The size of the  $w$  window is crucial for capturing the image features of a desired scale. As shown in (1), the tensor gives three features for

one pixel. Moreover, the resulting  $S(p)$  matrix contains information about the direction and strength of the edges in the neighbourhood of the  $p$  point. The two eigenvectors of the matrix are aligned orthogonally and parallel to the local edge (intensity change) in the windows  $w$ , while the corresponding eigenvalues indicate the strength of the edge in both directions.

### 2.3 Level sets and the ACWE model

An active contour, according to the level set idea [37], can be defined as a curve  $C$  in a form of a set of zero-height points  $p = (x, y)$  on the surface  $\phi(p, t)$ . This formulation gives  $C = \{p : \phi(p, t) = 0\}$ , where  $\phi(p, t) : \mathfrak{R}^2 \mapsto \mathfrak{R}$  and  $t$  is the evolution time step. The curve evolves in its normal direction according to the differential equation:

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| F, \quad \phi(p, 0) = \phi_0(p), \tag{2}$$

where  $\phi_0$  is the initial contour and  $F$  is a speed function  $F(p, t)$ , which allows the contour to expand or contract in order to encompass the segmented region. While the originally proposed level-set based methods mostly relied on edge information to drive the contour [6, 7], Chan and Vese proposed a global region based ACWE model [9] inspired by the Mumford–Shah formulation of the segmentation problem [35]. Given an image  $I$ , defined in the  $\Omega$  domain, a piecewise constant segmentation of  $I$  into foreground and background can be obtained with the curve  $C$ , evolved by minimising the ACWE energy functional defined as:

$$\begin{aligned} F(c^+, c^-, C) = & \mu \cdot Length(C) \\ & + \lambda_1 \int_{inside(C)} |I(p) - c^+|^2 dx dy \\ & + \lambda_2 \int_{outside(C)} |I(p) - c^-|^2 dx dy, \end{aligned} \tag{3}$$

where  $c^+$  and  $c^-$  represent the average pixel intensities of  $I$  inside and outside of the curve  $C$ , respectively,  $I(p)$  is the image intensity in the point  $p$ ,  $Length(C)$  is the total length of the curve, and  $\mu \geq 0$  and  $\lambda_1, \lambda_2 > 0$  are positive constant weights. This functional aims to minimise the intensity variance inside and outside the contour.

This minimisation problem can then be redefined using the level set approach. The curve represented as a zero level set function  $C = \{p \in \Omega \mid \phi(p) = 0\}$ , where  $\phi(p) > 0$  is the inside of the curve and  $\phi(p) < 0$  is outside. With this curve formulation, the energy functional takes the form:

$$\begin{aligned} F(c^+, c^-, \phi) = & \mu \cdot \int_{\Omega} \delta(\phi(p)) |\nabla \phi(p)| dx dy \\ & + \lambda_1 \int_{\Omega} |I(p) - c^+|^2 H(\phi(p)) dx dy \\ & + \lambda_2 \int_{\Omega} |I(p) - c^-|^2 (1 - H(\phi(p))) dx dy, \end{aligned} \tag{4}$$

where  $H(z)$  and  $\delta(z)$  are regularised versions of 1D Heaviside and Dirac delta functions, defined as:

$$H(z) = \begin{cases} 1 & \text{if } z \geq 0, \\ 0 & \text{if } z < 0, \end{cases} \quad \delta(z) = \frac{d}{dz} H(z). \tag{5}$$

To finally solve the problem, the Euler–Lagrange equation can be obtained by minimizing the energy with respect to  $\phi$  and parameterising the descent direction by the evolution time  $t$  while keeping  $c^+$  and  $c^-$  constant:

$$\frac{\partial \phi}{\partial t} = \delta(\phi) \left[ \mu \operatorname{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) - \lambda_1 (I - c^+)^2 + \lambda_2 (I - c^-)^2 \right], \tag{6}$$

where  $\operatorname{div}$  denotes the divergence operator. After each iteration,  $c^+$  and  $c^-$  are updated according to:

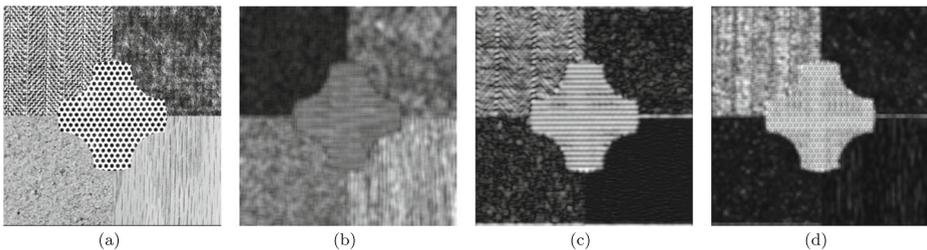
$$\begin{aligned} c^+ &= \frac{\int_{\Omega} I(p) H(\phi(p)) dx dy}{\int_{\Omega} H(\phi(p)) dx dy}, \\ c^- &= \frac{\int_{\Omega} I(p) (1 - H(\phi(p))) dx dy}{\int_{\Omega} (1 - H(\phi(p))) dx dy}. \end{aligned} \tag{7}$$

The initial ACWE model formulation was extended to handle vector-valued images [8]. Instead of two average intensity values (inside and outside the curve), the method uses vectors of the average intensities for each image channel. In this form, the image defined on  $\Omega$  consists of multiple channels  $I_i$ , where  $i = 1, \dots, N$ . The energy functional  $F_v(\bar{c}^+, \bar{c}^-, C)$  is then defined as:

$$\begin{aligned} F_v(\bar{c}^+, \bar{c}^-, C) &= \mu \cdot \operatorname{Length}(C) \\ &+ \int_{in(C)} \frac{1}{N} \sum_{i=1}^N \lambda_i^+ |I_i(p) - c_i^+|^2 dx dy \\ &+ \int_{out(C)} \frac{1}{N} \sum_{i=1}^N \lambda_i^- |I_i(p) - c_i^-|^2 dx dy, \end{aligned} \tag{8}$$

where  $\bar{c}^+ = (c_1^+, \dots, c_N^+)$  and  $\bar{c}^- = (c_1^-, \dots, c_N^-)$  are two vectors with the average values of the image channels inside ( $\bar{c}^+$ ) and outside ( $\bar{c}^-$ ) of the curve, and  $\lambda_i^+$  and  $\lambda_i^-$  are now separate weights for each channel.

The advantage of this model lies in its ability to separate image regions even without a sharp edge between them. This can be visible in the case of some texture features (see Fig. 1) when the separation can be possible, but the edges are blurred. The vector-valued version can also detect region dissimilarities that are present only in some of the channels.



**Fig. 1** Example texture feature maps generated by different methods: **a** source image, **b** GLCM Homogeneity, **c** Gabor filter magnitude, **d** first structure tensor eigenvalue

### 3 Method overview

The proposed segmentation method is based on a two-dimensional level set representation of an active contour. After a manual initialisation of the contour, the algorithm performs three main stages: initial texture feature extraction and selection, followed by two curve evolution stages (see Fig. 2). The first stage generates a set of texture feature maps that will be used to guide the further evolution of the contour. The next stage is based on a traditional level set propagating front formulation. The propagation speed function uses a strict textural stopping term that tracks sudden changes in the selected feature maps. Finally, the last stage switches to a variational level set framework based on vector-valued ACWE method. The second stage goal is to give a general outline of the segmented region, while the third stage is designed to provide a refined final form of the contour. The method also makes heavy use of GPU acceleration in most of the performance-critical computations.

#### 3.1 Texture feature extraction and selection

The method starts with a texture feature extraction and selection stage, similar to the method described in [42]. The stage results in a set of texture feature maps  $M_{best}$ , where a single feature map  $m$  is a scalar image that contains the feature values for each pixel of the original image. Any feature extraction method that is able to generate such a map can potentially be applied.

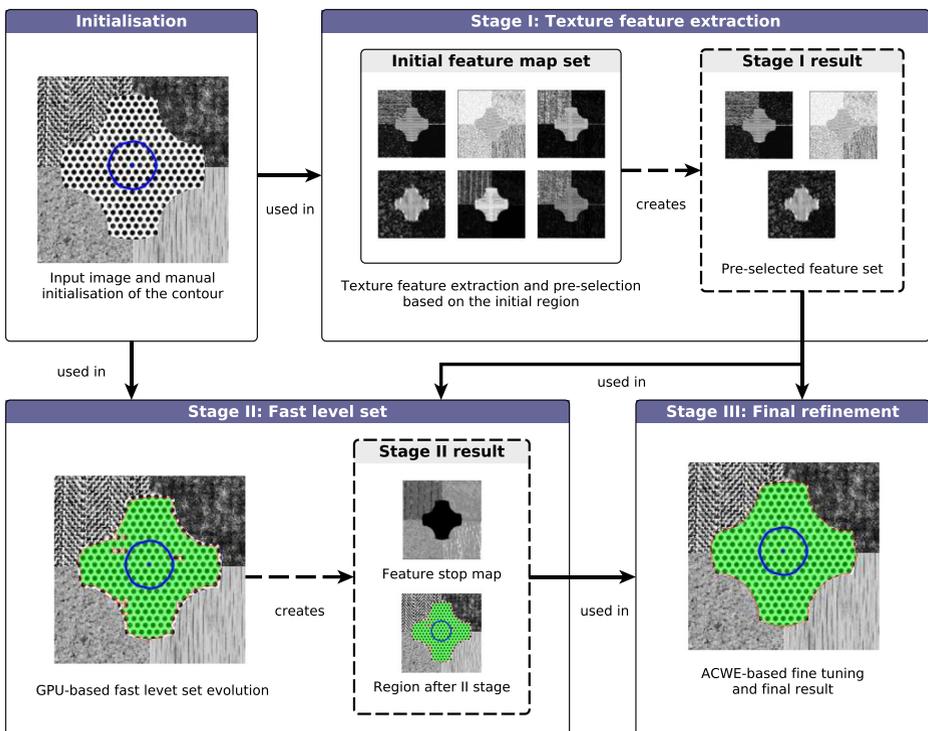


Fig. 2 Overview of the proposed method

The extraction starts with a list of all possible feature maps that can be generated with the available algorithms. Instead of calculating whole maps for the entire image, the process generates the  $M_{init}$  set of maps only in the bounding box of the initial contour. For best results, this initialisation should capture a representative sample of the pattern. The features are then pre-selected to the  $M_{best}$  set defined as:

$$M_{best} = \{m \in M_{init} : \%RSD(m) < r\}, \quad (9)$$

where  $r$  is the user-provided threshold (equal to 65% by default) and  $\%RSD(m)$  is the Relative Standard Deviation of feature map  $m$  inside the initial contour. In this way, the process step picks the maps with the most uniform feature values inside the contour, according to a specified threshold. This criterion assumes that the feature uniformity identifies it as a suitable descriptor of the initial pattern. Furthermore, its important change during the contour evolution can signal a change in the texture. Next, the feature maps in  $M_{best}$  are re-calculated for the entire image and are ready for the following stages.

The current implementation uses GLCM, Gabor and structure tensor features. As for GLCM, five features are calculated: Entropy, Correlation, Homogeneity, Contrast and Energy. A feature space is generated using a combination of specific parameters: window size, displacement and orientation. The extraction process starts with a fixed set of GLCM properties and, if necessary, performs additional computations. Automatic increase of the window size (from  $3 \times 3$ ,  $5 \times 5$  and  $7 \times 7$  upward) can be performed in the case of larger patterns and redundant channels for different angles can be omitted or merged in case of isotropic texture patterns. Please refer to [42] for more details on these operations. Moreover, a bank of 24 Gabor filters with four orientations ( $0^\circ$ ,  $45^\circ$ ,  $90^\circ$  and  $135^\circ$ ), three wavelengths and two Gaussian envelope scales is considered. The generation process also employs an angle detection scheme, similar as in the case of GLCM features. Finally, the method constructs a set of structure tensor features. This set consists of maps of both eigenvalues of tensors generated from  $\sigma$  and  $w$  combinations for a total of 18 feature maps.

### 3.2 Fast level set evolution

In the second stage, the initialised contour is deformed using the level set method. This stage is based on the traditional level set formulation presented in (2), where the speed function  $F$  in the point  $p$  combines image and curvature terms [27] as:

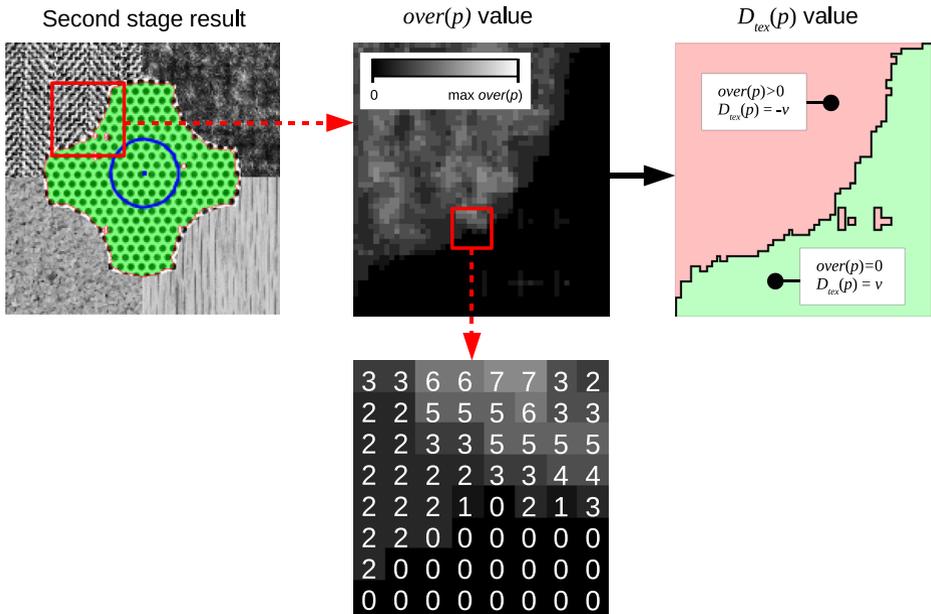
$$F(p, t) = \alpha D(p) + (1 - \alpha)C(p), \quad (10)$$

where  $D(p)$  is the image data term that drives the deformation,  $C(p) = \text{div}(\nabla\phi(p)/|\nabla\phi(p)|)$  is the curvature and  $\alpha \in [0, 1]$  is a user-defined balancing parameter. Instead of the original intensity-based image term, we apply a texture-based term  $D_{tex}(p)$  [41] that takes into consideration the previously selected texture feature set. First, for a given image point  $p$ , a set of features  $S_p$  is defined as:

$$S_p = \{m \in M_{best} : |m(p) - \bar{x}(m)| > \theta \cdot \sigma(m)\}, \quad (11)$$

where  $m$  is a texture feature map in the selected set  $M_{best}$ ,  $\bar{x}(m)$  and  $\sigma(m)$  are the feature map mean and standard deviation inside the initial contour,  $m(p)$  is the value of the map  $m$  in the point  $p$  and  $\theta$  is a user-defined parameter that denotes the term sensitivity. In this way, the  $S_p$  set will contain the features in point  $p$  that are sufficiently dissimilar to their mean values inside the initial contour. Next, a function  $over(p)$  is defined to return a number of features that overstep the condition in (11) for each point in the image  $I$ :

$$over(p) = |S_p|, \quad p \in I. \quad (12)$$



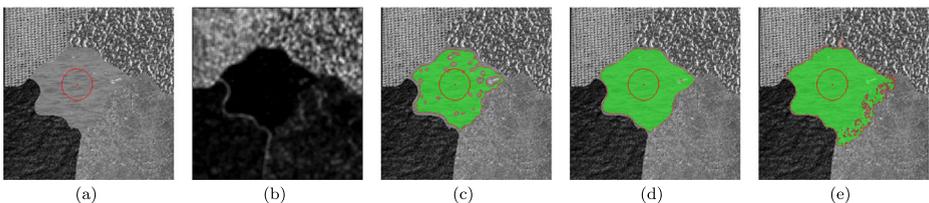
**Fig. 3** The second stage of the method illustrated. The  $over(p)$  function value in the given point, visualised in greyscale, is equal to the number of features that do not satisfy the similarity criterion

Finally, the texture data term is defined as:

$$D_{tex}(p) = \begin{cases} v & over(p) = 0 \\ -v & otherwise, \end{cases} \tag{13}$$

where  $v$  is a predefined constant. In this way, the curve is encouraged to expand into points where the texture features have values similar to the interior of the initial contour, but even one feature dissimilar enough will result in a strong penalty (see Fig. 3).

The second stage of the algorithm is expected to give a rough outline of the segmented region. An exact segmentation can be hard to achieve due to characteristic of some feature extraction methods [5, 20] that can provide distinctive, but blurry region boundaries. Figure 4c shows a result of this stage, where the contour covers most of the target region, but still contains some internal irregularities. Its edges are close to, but still not exactly on the desired boundaries of the region. This way of action is our deliberate choice: we assume



**Fig. 4** Multiple stages of the segmentation process: **a** initialisation, **b** image with pixel intensity corresponding to the  $over(p)$  function value, **c** second stage result, **d** final result, **e** oversegmentation example

that it is better to stop before passing the boundary than to cause an oversegmentation (see Fig. 4e). To encourage this behaviour, the default values of parameters were experimentally set to  $v = 20$ ,  $\alpha = 0.05$  and  $\theta = 4$ , where  $\theta$  is the main user-adjustable method sensitivity.

The level set equation is solved with a GPU-accelerated implementation of the traditional numerical scheme [37]. The simulation is performed until the convergence of  $\phi$  or a specified maximum number of iterations is reached.

### 3.3 Final contour refinement

The third stage is designed to provide a final form of the contour. At this point, the method switches from a classical level set algorithm to a region-based vector-valued ACWE model that will work on previously selected texture feature maps. Furthermore, the information about the texture dissimilarities from (11) influences the length term of the energy functional (8). Originally, the  $\mu$  parameter, along with the length contour  $\mu \cdot \text{Length}(C)$ , is used for scaling/regularising of the contour shape. Smaller values of  $\mu$  permit the contour to detect smaller objects in the image, at the cost of the contour smoothness.

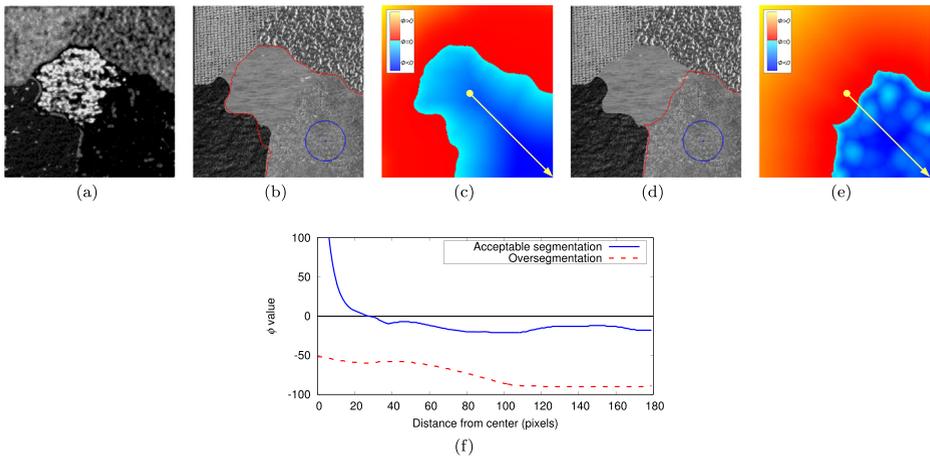
Conventionally, the  $\mu$  length weight is set constant for the entire evolution process [9]. In the proposed approach the  $\mu$  is scaled according to the previously obtained information. After the second phase, the  $over(p)$  function can give some information about the features that caused the contour stoppage in each point. In the proposed approach, the contour length term is redefined to include a normalised sum of the texture features that did not meet the similarity condition (11) in a given point. The new term  $\mu_{stop}(p)$  is defined as:

$$\mu_{stop}(p) = \frac{over(p)}{\max_{q \in I} over(q)} \cdot 100 \quad (14)$$

This term gives a different response in each of the image points. Its main idea is to generally discourage the contour from expanding to the areas with a large number of sufficiently different features. This penalty is also scaled according to the strength of the feature response function  $over(p)$ . Larger values of  $over(p)$  will result in a greater penalty, while lower values will encourage the contour to grow slowly, but still in a noticeable way. An example of the benefit of this length control term is shown in Fig. 5. The  $\mu_{stop}(p)$  term is visualised in a form of a “stop map” (Fig. 5a) for each of the image points and shows a strong discrimination between the central texture patch and the target region. Without the influence of this term the method clearly fails (Fig. 5b). The provided  $\phi$  heat maps and the plot along the marked line show that the improved length term helps to give a clear boundary between the regions.

### 3.4 Implementation concerns

The most computationally intensive parts of the method, like GLCM or Gabor feature generation and the second stage level set algorithm, were implemented using the GPU acceleration. In the optimal case, a GPU-running function (kernel) should be designed to work independently on its own part of the data and have the relatively small local memory requirements. In the case of the GLCM features, a local co-occurrence matrix has to be calculated in a window around each of the image pixels. To mitigate a potential memory problem, a quantisation is applied to reduce the matrix size. As for the Gabor features, the implementation uses a simple convolution of the original image with a given filter, which easily scales on a GPU. The second stage level set method is also well suited for the GPU, as each separate position on the level set function can be recalculated independently. The



**Fig. 5** Influence of the length term control: **a** visualisation of  $\mu_{stop}(p)$  function, **b** result with no length term control, **c**  $\phi$  heat map for the oversegmented result, **d** acceptable segmentation result with the corresponding  $\phi$  heat map **e** and **f** plot with the  $\phi$  value of both cases along the marked arrow

third ACWE-based stage is still CPU-based, but it usually requires only 5 to 15 iterations before the stopping criterion is reached.

The whole algorithm is implemented using the MESA system [43] – a platform for designing and evaluation of the deformable model-based segmentation methods. While the platform uses the Java language, the GPU-accelerated algorithms were written in C using OpenCL.

### 4 Experimental evaluation

The proposed method was tested on synthetic images created using the Brodatz texture dataset, as well as on a set of natural images. The initial contours were manually placed inside the desired regions and scaled to the preferred size. Unless specified otherwise, only the sensitivity parameter  $\theta$  was manually adjusted (between 4 and 12), while the other parameters were left on their default values. The first stage of the algorithm selected usually from 20 to 30 feature maps from 60 to 90 available.

The experiments were performed on a workstation with Intel Xeon E5-1620v2 CPU, 16 GB RAM, and Nvidia Titan Xp GPU. The total segmentation time was between 3 to 6 seconds for the analysed synthetic and natural image examples. As the implementation is heavily GPU-bound, other graphics cards were also tested.

The segmentation quality was assessed with five error measures: *overlap error* (OE), *area difference* (AD), *average symmetric surface distance* (ASSD), *Hausdorff distance* (HD), and *Dice Coefficient* (DC) [21]. Given two sets of image pixels  $S$  and  $G$ , where  $S$  is the tested result and  $G$  is the ground truth segmentation, the measures are defined as:

$$\begin{aligned}
 OE &= 100 (1 - |G \cap S|/|G \cup S|), \\
 AD &= 100 ((|S| - |G|)/|G|), \\
 DC &= 2 |G \cap S|/(|G| + |S|).
 \end{aligned}
 \tag{15}$$

DC and OE are popular overlap error measures. In contrast to these two, the AD measure does not factor the actual overlap of the sets but quantifies just their area difference. Together with OE and DC, however, it can indicate over- or undersegmentation of the results.

The ASSD takes into consideration the distances between the surfaces/borders of the sets (i.e. the voxels/pixel that have at least one background or edge point in their vicinity). For each point  $s_G$  in the border set  $S(G)$  the function  $d(s_G, S(R))$  denotes the Euclidean distance from  $s_G$  to the closest pixel in  $S(R)$ . These distances are also symmetrically calculated from the border pixels of  $R$  to  $G$ . All distance values are then averaged, which defines the ASSD as:

$$ASSD(G, R) = \frac{1}{|S(G)| + |S(R)|} \left( \sum_{s_G \in S(G)} d(s_G, S(R)) + \sum_{s_R \in S(R)} d(s_R, S(G)) \right) \quad (16)$$

In a similar fashion, the HD is the longest distance between any two points from the borders of the segmentation and ground truth regions (i.e. the greatest of all the distances from a point on one border the closest point in the other). With the border sets  $S(G)$  and  $S(R)$ , and  $d(g, r)$  as the distance between pixels  $g$  and  $r$ , the HD can be defined as:

$$HD(G, R) = \max \left\{ \sup_{g \in S(G)} \inf_{r \in S(R)} d(g, r), \sup_{r \in S(R)} \inf_{g \in S(G)} d(g, r) \right\}. \quad (17)$$

### 4.1 Synthetic images

The first example (see Fig. 6) presents segmentation the results of synthetically composed images. The method was able to successfully segment textures of different scales and orientations, even when multiple patterns were present. It should be noted that a more careful placement and scaling of the initial contour was necessary only in the last example. The experiments were also performed with the feature pre-selection criterion from (9) disabled. Normally, about 50 to 70% of the initial feature maps were pre-selected. The omission of this operation did not have a real influence on the segmentation results but caused an about 30% increase in the segmentation time.

The second example in Fig. 7 illustrates the influence of the  $\theta$  sensitivity parameter. The first case (Fig. 7a) shows the second stage result with  $\theta$  set to the default value of 4, where

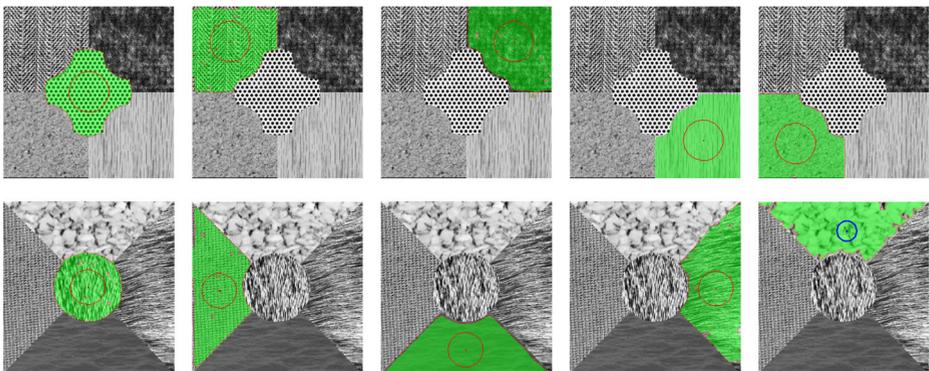
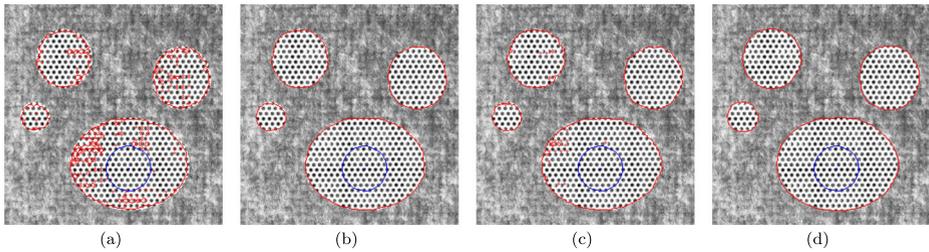


Fig. 6 Segmentation of synthetic texture mosaics

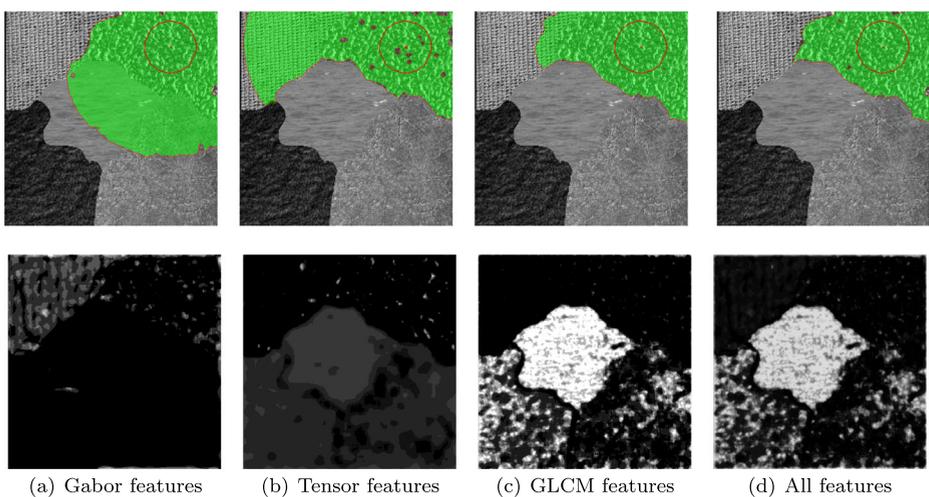


**Fig. 7** Influence of the  $\theta$  sensitivity parameter on the intermediate and final segmentation result: **a** result of second phase with  $\theta = 4$ , **b** result of third phase with  $\theta = 4$ , **c** second phase result for  $\theta = 7$  and **d** third phase result for  $\theta = 7$

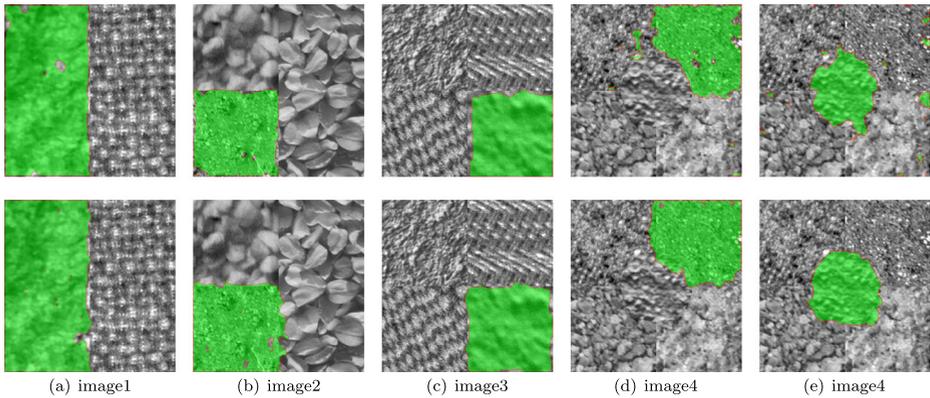
the contour exhibits some significant gaps, but the third stage provides an acceptable final result (Fig. 7b). The result of the second stage with  $\theta = 7$  is closer to the desired form, but in both of the cases, the final result is comparable. Generally, the  $\theta$  parameter permits a wide range of values that can give an acceptable segmentation.

The third example contains a particularly difficult case (Fig. 8), where different types of features were necessary to segment the target region. On their own, the GLCM, Gabor and tensor features were not able to provide a strong boundary between all of four adjacent regions, but together they were able to perform a successful segmentation.

The next example compares the proposed method with a deep learning-based approach that uses fully convolutional networks for the task of image segmentation (FCNT), proposed by Andrearczyk and Whelan [2]. The experiments were performed on mosaics created from the Kylberg texture dataset [26], and the segmentation was compared with the results presented in [2]. Figure 9 contains selected outcomes of both algorithms, while complete quantitative quality comparison is presented in Table 1. The tested mosaics contain quite challenging patterns, but both methods are mostly able to provide reasonable segmentation, indicated by low VOE and RVD, as well as DC greater than 0.95 in many cases. The



**Fig. 8** Segmentation results with **a-c** only the specific features enabled and **d** with all features used (result regions in the top row and  $over(p)$  function value map in the bottom)



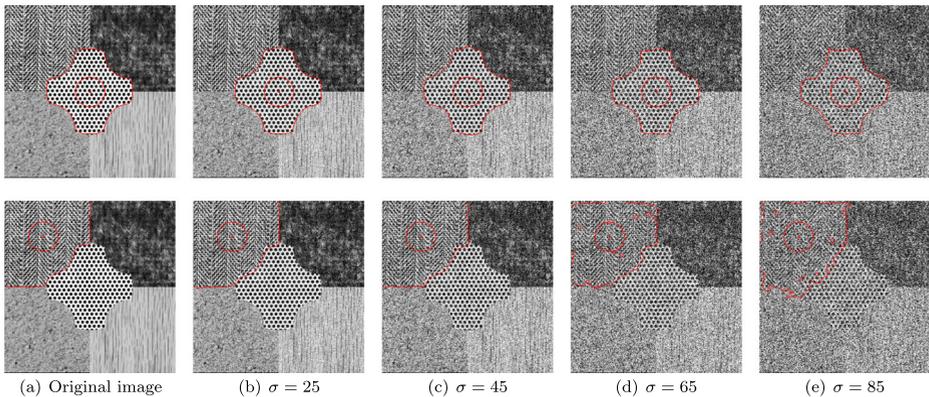
**Fig. 9** Segmentation results on mosaics from the Kylberg dataset: results of the FCNT method (top row) and the proposed algorithm (bottom row)

border-based metrics (ASSD and HD) show the biggest differences in favour of the proposed method. Some of the FCNT segmentations contain many disjointed patches outside of the target region, which are not present in the results of the proposed method. This is indicated by lower ASSD and significantly reduced HD (especially in Fig. 9d and Fig. 9e).

The last example (see Fig. 10) presents the influence of image noise on the performance of the proposed method. The ability to deal with noisy images is a desirable feature of robust segmentation methods, especially in medical applications [13], where input noise is common. The test image in the example was modified by the addition of Gaussian noise with increasing value of its standard deviation ( $\sigma$  from 45 to 85). Although the original

**Table 1** Accuracy of the Kylberg dataset mosaics segmentation performed with the FCNT and the proposed method

Image	Pattern	FCNT					Proposed method				
		VOE	RVD	ASSD	HD	DC	VOE	RVD	ASSD	HD	DC
image1	left	4.29	-4.21	3.52	71	0.98	<b>2.71</b>	<b>-2.18</b>	<b>2.33</b>	<b>70</b>	<b>0.99</b>
	right	3.07	<b>1.56</b>	3.66	99	0.98	<b>2.65</b>	2.21	<b>1.39</b>	<b>71</b>	<b>0.99</b>
image2	top left	17.72	-14.13	9.23	118	0.9	<b>11.73</b>	<b>4.48</b>	<b>3.72</b>	<b>33</b>	<b>0.94</b>
	bottom left	<b>6.13</b>	<b>-4.45</b>	<b>2.92</b>	<b>41</b>	<b>0.97</b>	7.48	6.45	3.44	45	0.96
image3	top left	<b>4.92</b>	-3.7	3.46	70	<b>0.97</b>	8.8	<b>0.73</b>	<b>3.3</b>	<b>20</b>	0.95
	top right	<b>6.4</b>	<b>0.68</b>	7.18	172	<b>0.97</b>	15.66	-1.74	<b>5.83</b>	<b>22</b>	0.92
	bottom left	<b>5.96</b>	<b>1.22</b>	<b>2.41</b>	26	<b>0.97</b>	9.98	2.39	4.28	<b>46</b>	0.95
	bottom right	8.41	-8.29	2.97	12	0.96	<b>5.71</b>	<b>-3.11</b>	<b>1.99</b>	<b>9</b>	<b>0.97</b>
image4	top left	10.58	-5.46	4.57	102	0.94	<b>10.18</b>	<b>2.16</b>	<b>2.32</b>	<b>16</b>	<b>0.95</b>
	top right	<b>14.24</b>	<b>7.56</b>	8.15	100	<b>0.92</b>	14.74	10.13	<b>3.33</b>	<b>18</b>	<b>0.92</b>
	bottom left	13.34	-3.18	3.95	53	0.93	<b>8.85</b>	<b>2.69</b>	<b>1.83</b>	<b>14</b>	<b>0.95</b>
	bottom right	22.16	<b>-10.14</b>	10.96	99	<b>0.88</b>	<b>21.95</b>	-16.49	<b>5.67</b>	<b>36</b>	<b>0.88</b>
	center	27.53	<b>8.6</b>	16.72	94	0.84	<b>17.01</b>	10.76	<b>3.32</b>	<b>15</b>	<b>0.91</b>



**Fig. 10** Influence of image noise on the segmentation results: **a** segmentation of two regions performed on unmodified images and **b–e** results on images with added Gaussian noise ( $\sigma$  values from 25 to 85)

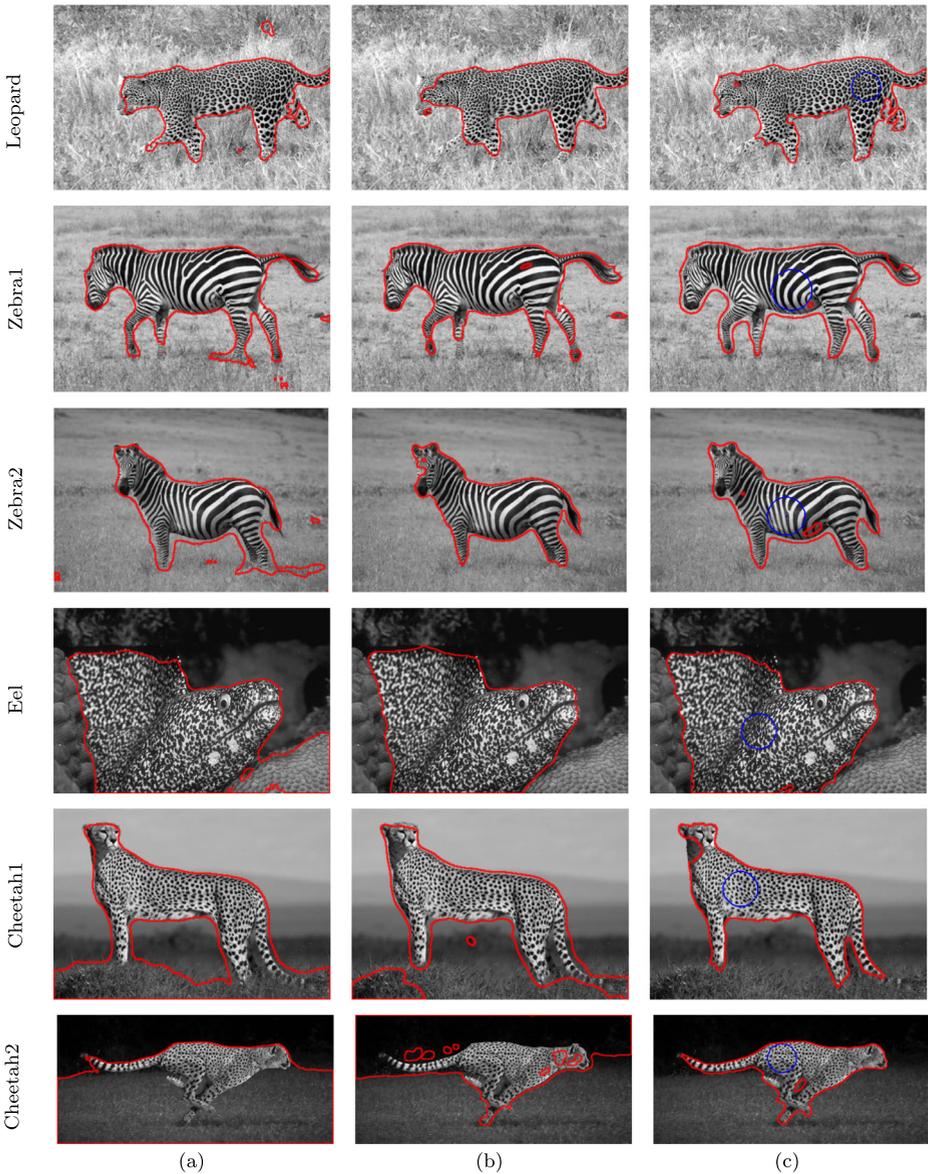
patterns significantly deteriorated on the most noisy images, the proposed method managed to perform adequately. Each increase in the  $\sigma$  value required only a slight decrease (by 0.5 to 1.0) in the  $\theta$  sensitivity parameter (starting from  $\theta = 6.5$  for the central pattern and  $\theta = 4$  for the top left region).

## 4.2 Natural images

Figure 11 contains the segmentation results on various natural images. The results of the proposed algorithm were compared with two state-of-the-art texture segmentation methods: Dictionary-based Active Contour (DAC) [11] and Factorisation-based Active Contour (FAC) [16]. Both methods employ a region-based level set framework but take different approaches to texture feature extraction. DAC constructs a dictionary of texture patches present in the segmented image. Each image patch is then assigned to an element of the dictionary. Using the initial form of the contour, the pixels in the dictionary patches are labelled as being inside or outside of the contour. This labelling is then used to calculate the probability of being inside the target region for each image pixel. The level set evolution process expands the curve into regions where the probability is high and retracts it from low probability areas. The FAC method uses a local spectral histogram [28] as the texture descriptor. After the feature extraction, the image is modelled as a product of a matrix with the features values and a matrix of the features weights for the segmented regions. FAC assumes two regions: foreground and background. During the contour evolution, the level set framework uses a factorisation algorithm to minimise the differences between the model (features and weights matrices) and the original image (real values of the extracted features).

The DAC and FAC results were obtained with MATLAB implementations shared by the authors. For the FAC, the parameters were set to 12 histogram bins and the integration scale of 20. The default circular initialisation was also used. As for the DAC method, the initialisation corresponded with the initial contours used by the presented method, with the suggested patch sizes from 9 to 15. The quantitative analysis of the segmentation quality is presented in Table 2.

The first three natural images (Leopard, Zebra1, and Zebra2) are more bi-modal in nature. For the Leopard image, the proposed method performance is on par with the other two algorithms. All methods achieved the DC around 0.93 and the OE values were between



**Fig. 11** Natural image segmentation with different methods: **a** FAC, **b** DAC and **c** the proposed method

11 and 12.5, with the lowest AD of -1.58 for the proposed method. The algorithms had the most problems in the bottom parts of the regions, where the leopard's limbs blend into the grass. The images with Zebras contain high-contrast patterns of variable scale. Here, the results of the proposed method are also comparable to DAC and FAC, although a slight oversegmentation can be noticed (AD = 13.65 for Zebra1 and AD = 14.75 for Zebra2).

In the next image (Eel) the scene is composed of multiple objects with different patterns. All methods were able to encompass most of the target region, with FAC also including a

**Table 2** Segmentation accuracy of the tested methods for natural images

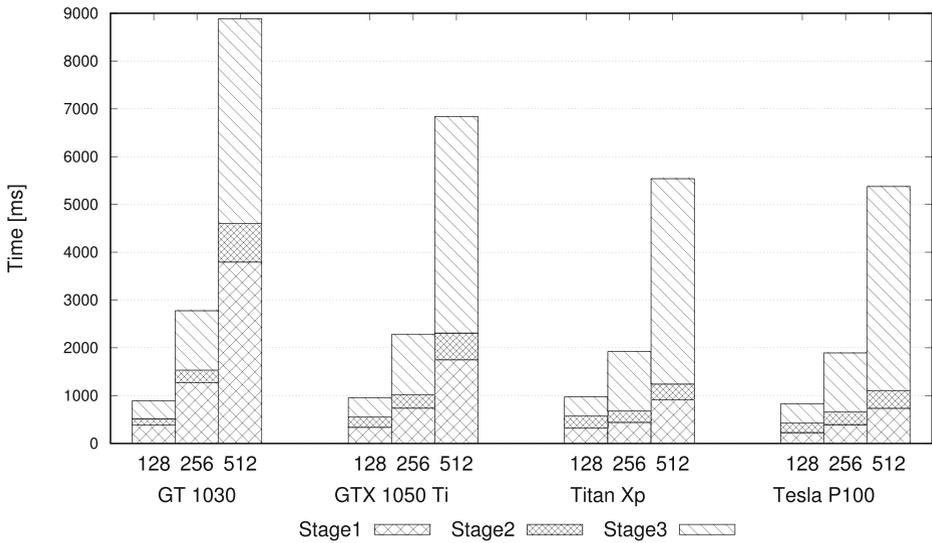
Image	Method	OE	AD	ASSD	HD	DC
Leopard	FAC	12.43	-5.28	4.38	65	0.93
	DAC	13.52	-8.86	3.68	38	0.93
	Proposed	11.50	-1.58	3.22	29	0.94
Zebra1	FAC	19.99	22.06	11.13	153	0.89
	DAC	7.50	2.93	1.63	13	0.96
	Proposed	13.81	13.65	3.33	20	0.93
Zebra2	FAC	13.65	3.50	4.00	58	0.93
	DAC	9.67	1.13	2.93	52	0.95
	Proposed	17.68	14.75	3.76	16	0.90
Eel	FAC	18.46	16.83	13.8	141	0.90
	DAC	3.44	-0.80	1.90	11	0.98
	Proposed	5.30	-3.18	2.93	24	0.97
Cheetah1	FAC	44.63	75.24	16.38	102	0.71
	DAC	29.01	37.83	12.43	102	0.83
	Proposed	8.53	-0.58	1.74	20	0.96
Cheetah2	FAC	75.60	299.43	44.42	153	0.39
	DAC	66.99	182.88	23.15	100	0.50
	Proposed	9.05	4.96	1.66	14	0.95

part of a different object. The other two methods achieved a high DC (0.97 and 0.98) with low values of OE, AD, and ASSD.

The last two images (Cheetah1 and Cheetah2) contain multiple texture patterns. Apart from the target object, there are at least two other background regions. The proposed method performed significantly better than FAC and DAC, especially in the case of the second image (Cheetah2). Since DAC and FAC target bi-modal images, the proposed method gave better results thanks to the employment of multiple features. As the previous examples show, the method can differentiate between distinct regions even without an explicit multi-phase formulation. It should be noted that there is also a multi-phase version of the DAC algorithm that could perform better in these conditions. However, due to the nature of the proposed method, comparison with other two-phase methods seemed more appropriate.

### 4.3 Time performance

Finally, the performance results of the method on GPUs of different classes are presented in Fig. 12. Four Nvidia cards were tested: low-end GT 1030, mid-range GT 1050 Ti, high-end Titan Xp, and server-grade Tesla P100 (please see Table 3 for specifications). The execution time is shown as a stacked chart of three stages of the algorithm. The tests were performed on three versions of the first synthetic image from Fig. 6. The image, originally of resolution  $256 \times 256$ , was also scaled to  $128 \times 128$  and  $512 \times 512$  pixels. The first stage of the algorithm was fixed to use GLCM features, since their extraction is the most computationally intensive. In the case of high-end cards, the provided example did not pose a significant challenge



**Fig. 12** Performance of the proposed method on different GPUs

(less than 1, 2, and 6 s for the three test images). The CPU-based third stage took the most time for the larger image sizes. Two lower-end cards were the more realistic targets for a 2D segmentation task. In their case, the achieved times were still reasonable. Even on the low-end GT 1030, the execution took less than 10 s for the largest image and was below 1 s and 3 s for  $128 \times 128$  and  $256 \times 256$  images, respectively.

The proposed method benefits significantly from the GPU acceleration. For example, the level set evolution in the second stage executes in less than 0.5 s in most of the performed tests. In contrast, a CPU-based version can take from 4 to 8 s to finish on a  $256 \times 256$  image and up to 25 s in the  $512 \times 512$  case. The GPU-based GLCM algorithm also exhibits similar improvements in execution time.

#### 4.4 Discussion

The proposed method was compared against other active contour-based methods, as well as against a deep learning-based approach. Despite the fact that the presented algorithm essentially works in a two-phase mode, it can clearly perform well on multi-modal images with multiple textures. The relevant multiple texture features can complement each other in those situations, while the unused descriptors are not getting in the way of the final results.

**Table 3** Specifications of the graphics cards used for performance tests

Model	CUDA Cores	RAM [GB]	Clock [MHz]
GT 1030	384	2	1468
GT 1050 Ti	768	4	1392
Titan Xp	3840	12	1582
Tesla P100	3584	16	1303

The GPU-based implementation provides a clear improvement in the runtime performance. Historically, algorithms like level sets and GLCM were notoriously computationally intensive, which limited their practical applicability. The experiments show that modern GPUs can compute relatively large banks of texture features and perform the “fast” level set evolution in sub-second time, which invalidates some previous restrictions. The GPU-based level set method in the second stage is driven by a speed function with a relatively small influence of the curvature, and the parameters of the evolution are tuned for possibly fast convergence. Furthermore, the computations are performed on the entire image domain. Typical optimisations aim to reduce the range of the level set function updates with sparse schemes [27], where only a part of the image domain is considered (e.g. only a narrow band around the level set zero-crossing is modified in each iteration). These modifications can fundamentally change the final results, while our approach is clearly efficient enough, at least in the 2D case.

The comparison with the FCNT, which uses fully convolutional networks, indicates that the proposed method can be competitive against current state-of-the-art approaches. Deep learning algorithms provides some advantages over active contours and level sets, like the lack of necessity for manual initialisation and interactive parameter tuning. On the other hand, the (often supervised) training stage is unavoidable and depends on the availability of prepared and annotated datasets.

The presented method employs a multi-stage approach that was firstly outlined in [42]. The original work used a snake-based ACM that employed a vastly different evolution scheme that considered only the closest neighbourhood of the curve discrete points. The proposed approach uses both local and global level set-based ACMs that are more robust in handling of multiple features. The topological restrictions of the parametric snake are also eliminated (see Fig. 7). Furthermore, the final refinement stage is not only different from the heavily heuristic approach of the snake version, but also uses a distinct, region-based level set model than the previous stage. This cooperation of those two models is one of the main features of the proposed method. The computational advantage of the discrete snake was also nullified with the GPU utilisation.

## 5 Conclusions and future works

In this paper, a multi-stage texture-based active contour has been presented. The proposed method combined both a classical and region-based level set contour formulations and smartly integrated multiple and varied texture features. The algorithm was carefully validated on synthetic and natural images and compared with the most similar among the available and accessible state-of-the-art approaches. The method was able to successfully segment various patterns in case of bi- and multi-modal images. It should be noticed that the presented approach is not tied to any particular texture descriptor and can be easily extended to integrate additional features. Moreover, it efficiently employs a large number of different kinds of descriptors thanks to the GPU acceleration.

We are aware that the current form of the method can be improved in many ways. Although the algorithm does not require any prior information about the texture classes in the image, it relies on manual contour initialisation. A more advanced feature selection scheme could also be employed in the first stage. As for the performance, the third ACWE-based stage would benefit greatly from GPU acceleration. Moreover, the general level set form of the algorithm could be extended into a multi-phase version that could segment multiple regions [3].

Now, we are working on an adaptation of the proposed segmentation method into 3D [18, 27]. We are also considering assessing the method on biomedical images and more experimental comparisons with very popular, fully convolutional approaches [29, 54]. Finally, we plan to tackle a more complex task of semantic segmentation.

**Acknowledgments** This work was supported by the Polish National Science Centre under Grant No. 2017/25/N/ST6/01849. The authors are grateful to Dr. Cezary Boldak for his contribution to the work

## Compliance with Ethical Standards

**Conflict of interests** The authors declare that they have no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Ahmad A, Badshah N, Ali H (2020) A fuzzy variational model for segmentation of images having intensity inhomogeneity and slight texture. *Soft Computing* 24:15491–15506
- Andrearczyk V, Whelan PF (2017) Texture segmentation with fully convolutional networks. *arXiv:1703.05230*
- Awate SP, Tasdizen T, Whitaker RT (2006) Unsupervised texture segmentation with nonparametric neighborhood statistics. In: *Computer Vision – ECCV 2006*, pp. 494–507. Springer
- Bigün J (1987) Optimal orientation detection of linear symmetry. In: *First International Conference on Computer Vision, ICCV*, pp. 433–438. IEEE
- Bigün J, Granlund G, Wiklund J (1991) Multidimensional orientation estimation with applications to texture analysis and optical flow. *IEEE Trans. Pattern Anal. Mach. Intell.* 8:775–790
- Caselles V, Catté F, Coll T, Dibos F (1993) A geometric model for active contours in image processing. *Numer Math* 66(1):1–31
- Caselles V, Kimmel R, Sapiro G (1997) Geodesic active contours. *Int. J. Comput. Vis.* 22(1):61–79
- Chan T, Sandberg B, Vese L (2000) Active contours without edges for vector-valued images. *J. Vis. Commun. Image Represent.* 11(2):130–141
- Chan T, Vese L (2001) Active contours without edges. *IEEE Trans. Image Process.* 10(2):266–277
- Cohen LD (1991) On active contour models and balloons. *CVGIP: Image Understanding* 53:211–218
- Dahl A, Dahl V (2015) Dictionary based image segmentation. In: *Scandinavian conference on image analysis*, pp. 26–37. Springer
- Daugman J (1985) Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J. Opt. Soc. Am. A* 2(7):1160–1169
- Diwakar M, Kumar M (2018) A review on CT image noise and its denoising. *Biomedical Signal Processing and Control.* 42:73–88
- Dong Y, Zhang H, Liu Z, Yang C, Xie GS, Zheng L, Wang L (2019) Neutrosophic set transformation matrix factorization based active contours for color texture segmentation. *IEEE Access* 7:93887–93897
- Esedoglu S, Ruuth S, Tsai R (2005) Threshold dynamics for shape reconstruction and disocclusion. In: *Proc. IEEE Inter. Conf. on Image Processing*, vol 2, pp 502–505
- Gao M, Chen H, Zheng S, Fang B (2016) A factorization based active contour model for texture segmentation. In: *Proc. IEEE Inter. Conf. on Image Processing*, pp. 4309–4313. IEEE
- Gao M, Chen H, Zheng S, Fang B (2019) Feature fusion and non-negative matrix factorization based active contours for texture segmentation. *Signal Process* 159:104–118
- Grushnikov A, Niwayama R, Kanade T, Yagi Y (2018) 3D level set method for blastomere segmentation of preimplantation embryos in fluorescence microscopy images. *Mach. Vis. Appl.* 29(1):125–134

19. Han B, Wu Y (2020) Active contour model for inhomogenous image segmentation based on Jeffreys divergence. *Pattern Recogn.* 107(107520)
20. Haralick R, Shanmugam K, Dinstein I (1973) Textural features for image classification. *IEEE Trans. Syst. Man Cybern. Syst.* 6:610–621
21. Heimann T, van Ginneken B, Styner MA et al (2009) Comparison and evaluation of methods for liver segmentation from CT datasets. *IEEE Trans. on Med. Imaging* 28:1251–1265
22. Houhou N, Thiran J, Bresson X (2008) Fast texture segmentation model based on the shape operator and active contour. In: *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern. Recognit.*, pp. 1–8
23. Huang X, Qian Z, Huang R, Metaxas D (2005) Deformable-model based textured object segmentation, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, 119–135
24. Jain A, Farrokhnia F (1991) Unsupervised texture segmentation using Gabor filters. *Pattern Recogn* 24(12):1167–1186
25. Kass M, Witkin A, Terzopoulos D (1988) Snakes: Active contour models. *Int. J. Comput. Vis.* 1(4):321–331
26. Kylberg G (2011) The Kylberg Texture Dataset v. 1.0 External report (Blue series), 35, Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, Uppsala, Sweden
27. Lefohn AE, Cates JE, Whitaker RT (2003) Interactive, GPU-based level sets for 3D segmentation R.e. ellis, T.M. Peters (eds.) *Proc. Med. Image. Comput. Assist. Interv. (MICCAI)*, pp. 564–572. Springer
28. Liu X, Wang D (2006) Image and texture segmentation using local spectral histograms. *IEEE Trans. Image Process.* 15(10):3066–3077
29. Lu X, Wang W, Ma C, Shen J, Shao L, Porikli F (2019) See more, know more: unsupervised video object segmentation with Co-Attention siamese networks. 2019 IEEE/CVF conference on computer vision and pattern recognition CVPR long beach, CA, USA, 3618–3627
30. McInerney T, Terzopoulos D (1999) T-snakes: Topology adaptive snakes. In: *Medical Image Analysis*, pp. 840–845
31. Mewada H, Patel R, Patnaik S (2014) A novel structure tensor modulated Chan–Vese model for texture image segmentation. *The Computer Journal* 58(9):2044–2060
32. Min H, Jia W, Wang X, Zhao Y, Hu R, Luo Y, Xue F, Lu J (2015) An intensity-texture model based level set method for image segmentation. *Pattern Recogn* 48(4):1547–1562
33. Moallem P, Tahvilian H, Monadjemi S (2016) Parametric active contour model using Gabor balloon energy for texture segmentation. *SIViP* 10(2):351–358
34. Moore P, Molloy D (2007) A survey of computer-based deformable models. In: *International Machine Vision and Image Processing Conference*, pp 55–66
35. Mumford D, Shah J (1989) Optimal approximations by piecewise smooth functions and associated variational problems. *Commun. Pure Appl. Math.* 42(5):577–685
36. Ni K, Bresson X, Chan T, Esedoglu S (2009) Local histogram based segmentation using the Wasserstein distance. *Int. J. Comput. Vis.* 84(1):97–111
37. Osher S, Sethian J (1988) Fronts propagating with curvature-dependent speed: algorithms based on hamilton-Jacobi formulations. *J. Comput. Phys.* 79(1):12–49
38. Paragios N, Deriche R (2002) Geodesic active regions and level set methods for supervised texture segmentation. *Int. J. Comput. Vis.* 46(3):223–247
39. Pujol O, Radeva P (2004) Texture segmentation by statistical deformable models. *Int. J. of Image and Graphics* 4(03):433–452
40. Reed T, DuBuf J (1993) A review of recent texture segmentation and feature extraction techniques. *CVGIP: Image Understanding* 57(3):359–372
41. Reska D, Boldak C, Kretowski M (2016) Toward texture-based 3D level set image segmentation, IP&C’15, Bydgoszcz, Poland. *Image processing and communications challenges 7. Advances in Intelligent Systems and Computing*, vol. 389: 205–211
42. Reska D, Boldak C, Kretowski M (2017) Towards multi-stage texture-based active contour image segmentation. *SIViP* 11(5):809–816
43. Reska D, Jurczuk K, Boldak C, Kretowski M (2014) MESA: Complete Approach for design and evaluation of segmentation methods using real and simulated tomographic images. *Biocybernetics and Biomedical Engineering* 34(3):146–158
44. Ronfard R (1994) Region-based strategies for active contour models. *Int. J. Comput. Vis.* 13(2):229–251
45. Rousson M, Brox T, Deriche R (2003) Active unsupervised texture segmentation on a diffusion based feature space. In: *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern. Recognit.*, vol 2, pp 699–704
46. Sagiv C, Sochen N, Zeevi Y (2006) Integrated active contours for texture segmentation. *IEEE Trans. Image Process.* 15(6):1633–1646

47. Sethian JA (1999) Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science, vol. 3 Cambridge University Press
48. Shen T, Zhang S, Huang J, Huang X, Metaxas D (2011) Integrating shape and texture in 3D deformable models: from Metamorphs to Active Volume Models. In: Multi modality state-of-the-art medical image segmentation and registration methodologies, pp. 1–31. Springer
49. de Siqueira F, Schwartz W, Pedrini H (2013) Multi-scale gray level co-occurrence matrices for texture description. *Neurocomputing* 120:336–345
50. Sochen N, Kimmel R, Malladi R (1998) A general framework for low level vision. *IEEE Trans. Image Process.* 7(3):310–318
51. Zhao G, Qin S, Wang D (2018) Interactive segmentation of texture image based on active contour model with local inverse difference moment feature. *Multimedia Tools and Applications* 77:24537–24564
52. Subudhi P, Mukhopadhyay S (2018) A novel texture segmentation method based on co-occurrence energy-driven parametric active contour model. *SIViP* 12(4):669–676
53. Tatu A, Bansal S (2015) A novel active contour model for texture segmentation. In: Energy min. Meth. Comput. Vis. Pattern. Recogn., pp. 223–236. Springer
54. Wang W, Lu X, Shen J, Crandall D, Shao L (2019) Zero-Shot Video object segmentation via attentive graph neural networks. In: Proc. IEEE Int. Conf. on Computer Vision, 9235–9244
55. Wang Y, Wang H, Xu Y (2013) Texture segmentation using vector-valued Chan–Vese model driven by local histogram. *Computers & Electrical Engineering* 39(5):1506–1515
56. Wang L, Zhang L, Yang X, Yi P, Chen H (2020) Level set based segmentation using local fitted images and inhomogeneity entropy. *Signal Process* 107297:167
57. Wu Q, Gan Y, Lin B, Zhang Q, Chang H (2015) An active contour model based on fused texture features for image segmentation. *Neurocomputing* 151:1133–1141
58. Xu C, Prince JL (1998) Snakes, shapes, and gradient vector flow. *IEEE Trans. Image Process.* 7(3):359–369
59. Yu H, He F, Pan Y (2020) A survey of level set method for image segmentation with intensity inhomogeneity. *Multimedia Tools and Applications*

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Daniel Reska** is a Ph.D. candidate in the Faculty of Computer Science at Bialystok University of Technology, Poland, where he received his MS degree in 2010. His research interests focus on image segmentation, medical image analysis, and parallel computing.



**Marek Kretowski** received a joint Ph.D. degree in 2002 from the Faculty of Computer Science at the Bialystok University of Technology, Poland, and from the University of Rennes 1, France. He is a Professor in the Faculty of Computer Science at the Bialystok University of Technology, Poland. His research interests focus on the biomedical applications of computer science (modelling for image understanding, image analysis), bioinformatics, and data mining.