IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 21, 2020, accepted July 16, 2020, date of publication August 4, 2020, date of current version August 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3014075

Multi-Resolution Texture-Based 3D Level Set Segmentation

DANIEL RESKA AND MAREK KRETOWSKI
Faculty of Computer Science, Bialystok University of Technology, 15-351 Białystok, Poland
Corresponding author: Daniel Reska (d.reska@pb.edu.pl)

This work was supported by the Polish National Science Centre under Grant 2017/25/N/ST6/01849.

ABSTRACT This article presents a novel three-dimensional level set method for the segmentation of textured volumes. The algorithm combines sparse and multi-resolution schemes to speed up computations and utilise the multi-scale nature of extracted texture features. The method's performance is also enhanced by graphics processing unit (GPU) acceleration. The segmentation process starts with an initial surface at the coarsest resolution of the input volume and moves to progressively higher scales. The surface evolution is driven by a generalised data term that can consider multiple feature types and is not tied to specific descriptors. The proposed implementation of this approach uses features based on grey level co-occurrence matrices and discrete wavelet transform. Quantitative results from experiments performed on synthetic volumes showed a significant improvement in segmentation quality over traditional methods. Qualitative validation using real-world medical datasets, and comparison with other similar GPU-based algorithms, were also performed. In all cases, the proposed implementation provided good segmentation accuracy while maintaining competitive performance.

INDEX TERMS Deformable models, GPU acceleration, image segmentation, level sets, texture analysis.

I. INTRODUCTION

Image segmentation can be defined as a process for partitioning the image into distinct regions. Nowadays, this task is often an important step in the analysis of both two-dimensional (2D) images and three-dimensional (3D) volumes. Applications such as autonomous driving, intruder detection, and medical diagnosis constantly pose new challenges for segmentation algorithms, demanding high quality results combined with reasonable performance. The efficiency of the segmentation methods can greatly benefit from acceleration by graphics processing units (GPUs) [1]–[3], which are inherently well equipped to handle parallel processing of 2D and 3D image data.

Deformable models [4] are a popular class of segmentation methods, based on the idea of a shape (a 2D curve or a 3D surface) that deforms in order to encompass a desired region. The shape evolution is usually driven by external image characteristics and internal forces (e.g. controlling the smoothness of the shape). Typically, edge information [5] or region intensity statistics [6], [7] are used as image properties for guiding a model's adaptation to the target area. For regions without clear borders or with non-uniform texture, however, these approaches are usually inadequate. To solve the problem, many texture feature extraction techniques, such as grey level co-occurrence matrices (GLCM) [8], [9], Gabor filters [10], matrix factorisation [11], [12], or wavelets [13], are integrated with deformable models. Typically, these methods use a single feature [11], [14], [15] or ad hoc combinations of descriptors [16], [17]. Moreover, the vast majority of the associated algorithms are limited to 2D images and are often designed with specific applications in mind [10], [18].

This research proposes a texture-based method for the segmentation of 3D volumes. The method uses a level set-based active surface model that evolves under the influence of 3D texture features. The model evolution employs a multi-resolution scheme [19] and sparse level set optimisations to speed up computations. The segmentation process starts at the coarsest resolution of the input volume and moves to progressively higher scales. The texture features are computed for each separate scale. In this article, we demonstrate a realisation of this approach, which utilised features based on GLCM and discrete wavelet transform (DWT) [20], [21]. GPU acceleration was used for the GLCM feature generation and the level set evolution. The experimental validation was performed on synthetically generated test volumes and real medical imaging datasets, for which both the quality of the segmentation and the computational performance

The associate editor coordinating the review of this manuscript and approving it for publication was Nilanjan Dey.

were analysed. The proposed method was also compared with state-of-the-art GPU-based methods.

The main contribution of this work is a new fast segmentation method that is not tied to specific texture features and can successfully be applied to diverse texture volumes. The high performance of the method was achieved, first, by the integration of multi-resolution and sparse approaches in both the level set and feature extraction methods and, second, its efficient implementation using a GPU.

This article is organised as follows. Section II provides a brief necessary background, followed by Section III, which describes the proposed method in detail. Section IV presents the experimental validation. The final section concludes the article and suggests avenues for possible future work.

II. RELATED WORKS

In this section, we present some background information regarding level set segmentation and GLCM- and DWT-based extraction of texture features.

A. LEVEL SETS AND 3D SEGMENTATION

Level set methods, originally designed to model propagating interfaces, are widely used in optimisation, computational geometry and image processing [22]. The idea of level set segmentation is based on an implicitly represented shape defined within an image domain. An active surface S can be defined as a set of zero-level points $p = (x, y, z)$ of the function $\phi(p, t)$. This formulation gives $S = \{p : \phi(p, t) = 0\}$, where $\phi(p, t) : \Re^3 \mapsto \Re$ and t is the evolution time step. The surface evolves in its normal direction according to the following partial differential equation (PDE):

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| F, \quad \phi(p, 0) = \phi_0(p), \tag{1}$$

where ϕ_0 is the initial surface and F is a speed function $F(p, t)$, which allows the surface to expand or contract in order to encompass the segmented region. Notable level-set based models include geometric [23] and geodesic active contours [24], as well as active contours without edges (ACWE) [6]. While the first two models rely on image edge information, the ACWE method takes a region-based approach, inspired by the Mumford–Shah formulation a segmentation problem [25]. The ACWE variational approach can provide a piecewise smooth partitioning of an image into separate regions, even if their edges are not well defined.

Level sets have an inherent ability to change their topology (divide or merge), which gives them an advantage over traditional parametric models [5]. The traditional explicit models, however, often have a computational advantage, since the computations can be limited to a relatively small number of contour control points. By contrast, solving a level set PDE using a finite difference-based method for an entire image domain, is much more computationally and memory intensive. Furthermore, the numerical stability of the method may be compromised due to irregularities that can appear during the evolution of the level set function. Periodic re-initialisation of the function is a common solution for this problem [22], but it may adversely affect the accuracy of the method and is time-consuming.

Many optimisations were introduced to address these problems. A *narrow band* approach can be used to limit the ϕ updates to only the region around a propagating contour [1], [26]. Schemes for sparse representations were also proposed [27]–[29], allowing improved processing and more space-efficient storage of the level set function. Furthermore, additional regularisation terms [30] and global minimisation schemes [31] were proposed to eliminate the need for the costly ϕ re-initialisation and improve the performance and accuracy.

Along with algorithmic improvements, the performance issues have been addressed by GPU utilisation. Rumpf and Strzodka [32] demonstrated an early application of GPU acceleration for the segmentation of 2D images, where the entire ϕ function was updated in each iteration. Lefohn *et al.* [29] proposed an optimised 3D method that used a sparse tile-based representation, which only processed the active regions around the zero-crossing of the level set function. The algorithm used a compaction scheme to transfer the data between the CPU and GPU.

Roberts *et al.* [1] created another notable sparse GPU-based algorithm. This solution used a narrow band approach, but tracking of the active computational domain was performed at the level of individual voxels. The number of coordinates processed at each time step was therefore minimised. The algorithm was used for the segmentation of 3D medical imaging datasets. A GPU-accelerated level set-based method was also proposed by Jalba *et al.* [33]. Their algorithm was designed for the task of surface reconstruction from point cloud data. The method achieved high performance due to a multi-scale tile-based level set implementation.

More recently, Willcocks *et al.* [34] presented a model that used a local Gaussian distribution fitting image term [35]. The model can segment regions with intensity non-homogeneities and can be resist noise. Although its GPU-accelerated algorithm does not use a sparse level set scheme, it performs efficiently enough to enable interactive parameter tuning and visualisation of the results.

B. GREY LEVEL CO-OCCURRENCE MATRIX FEATURES

A GLCM is a square matrix of order equal to the number of pixel grey levels. For a given spatial window of an image, the GLCM contains the probabilities of pair-wise occurrences of pixels with two grey levels. Each of the (i, j) matrix elements holds 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 used to calculate a set of texture feature descriptors, such as entropy, correlation, homogeneity, contrast or energy [8]. In the generation process, a set of features is usually selected and computed for a combination of parameters: window size, pixel distance and orientation. In the 2D case, 4 orientations are typically considered (see Fig. 1(a)), but in the 3D case [9] this

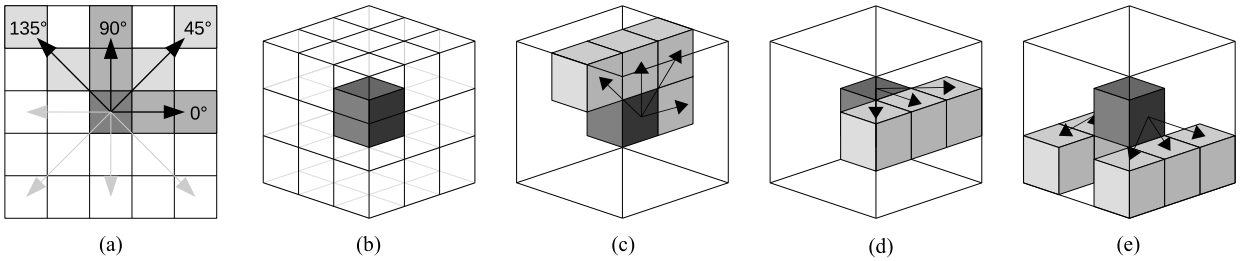

FIGURE 1. Illustrations of spatial relations during the calculations of GLCMs. Multiple directions and offsets can be used for a given pixel of interest. In the 2D case (a), four main directions are typically employed. In the 3D case with a voxel in the 3 × 3 × 3 neighbourhood (b), 13 directions can be defined for the adjacent voxels (c–e).

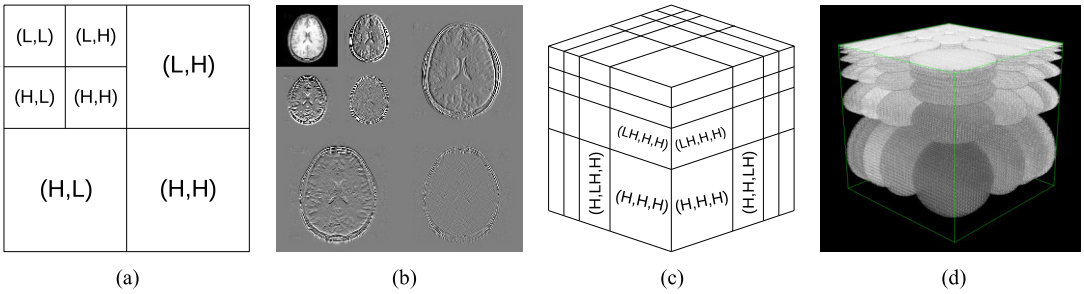

FIGURE 2. DWT decompositions of 2D and 2D data sets: (a) schema of 2-level 2D Mallat DWT decomposition with low-pass (L) and high-pass (H) filters, (b) coefficients of decomposition of a brain MRI image, (c) 3D 3-level hyperbolic DWT decomposition, and (d) visualisation of the coefficients of a hyperbolic decomposition of a spherical volume.

number can increase to 13 (see Figs. 1(b) to 1(e)). A GLCM is, therefore, a relatively computationally demanding method. It can also result in a high-dimensional space, requiring selection [15] or fusion [17] schemes.

C. DISCRETE WAVELET TRANSFORM

DWT creates a multi-resolution representation of an image that captures its spatial and frequency information. The processed image is recursively decomposed by being filtered by low-pass (L) and high-pass (H) filters along its dimensional direction. Fig. 2(a) shows a 2D Mallat [20] decomposition, with the image filtered by four combinations of H and L filters (horizontally, vertically, and diagonally). The H filters (HH, HL, and LH) give the *detail* wavelet coefficients, while the decomposition continues on the low-pass sub-band (the *approximation* coefficients LL). In this way, each level of decomposition gives a set of four bands of features.

A slightly different approach is used in hyperbolic wavelet transform (HWT). In this case, an image is first fully decomposed in each direction before moving to the next decomposition level. This approach is more computationally intensive but has the advantage of representation of anisotropic data [21]. Fig. 2(c) illustrates the HWT decomposition of a 3D volume. A visualisation of the absolute values of the coefficients is presented in Fig. 2(d).

III. THE PROPOSED METHOD

The proposed segmentation method is based on a 3D level set representation of an active surface. The surface evolution uses a multi-resolution approach [19]: the input volume is downscaled multiple times and the segmentation process spans the lower-resolution volumes until it ends at the original volume size (see Fig. 3). At each step, the level set method analyses the texture features generated for the current scale of the input. The algorithm is general in nature and not limited to any specific descriptors.

The method requires an initial form of the surface to be placed inside the target region. Next, this initialisation is used to calculate the starting form of the level set function ϕ, which evolves using the lowest resolution of the volume. At this resolution, the evolution ends when ϕ achieves convergence (the number of voxels added or removed from the segmented region drops below a specified threshold) or when a provided upper limit of iterations is reached. The resulting level set function is then up-sampled, reinitialised and evolves at a higher resolution. The progression scheme reduces the amount of computation required by the level set evolution. The algorithm naturally converges faster for a small input volumes. Furthermore, only the features relevant to the considered scale are calculated. By the time the highest volume resolution is reached, the bulk of the target region has already been segmented.

This process continues until the final scale is reached. At the original volume resolution, the level set method switches to a sparse evolution scheme that works only in the neighbourhood of the surface from the previous scale. The relevant blocks of the ϕ function are compacted and processed fully on the GPU. Furthermore, the texture features need to be generated only for the relevant regions of the image, resulting in a further reduction in computations.

A. TEXTURAL SPEED FUNCTION

At each resolution, the surface is deformed using the level set formulation from (1), in which the speed function F combines

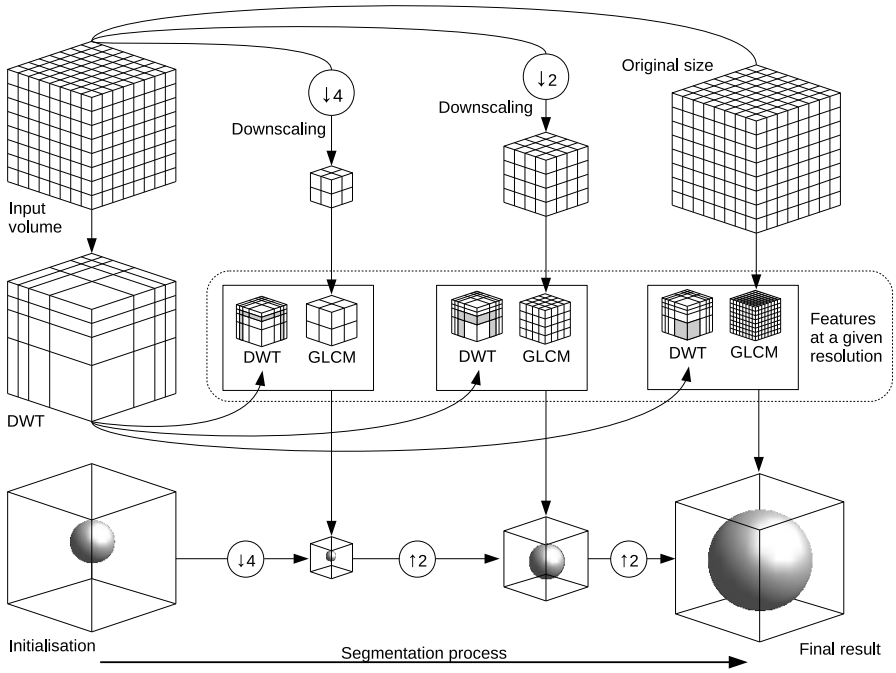

FIGURE 3. The stages of the proposed method. The segmentation process starts with the initial form of the surface (bottom left) and runs through downscaled versions of the input volume (top right) until it stops at the original resolution (bottom right). The employed texture features are generated separately for each scale of the input volume.

image and curvature terms for a point p:

$$F(p) = \alpha I(p) + (1 - \alpha)C(p), \tag{2}$$

where $I(p)$ is the image data term that drives the deformation, $C(p) = \text{div}(\nabla\phi(p)/|\nabla\phi(p)|)$ is the surface curvature and $\alpha \in [0, 1]$ is a user-defined balancing parameter. The original version [29] defined the data term as:

$$I(p) = \epsilon - |img(p) - T|, \tag{3}$$

where $img(p)$ is the image intensity in p, while T and ϵ are the intensity target value and tolerance: the surface is prompted to expand if $img(p)$ is between $T - \epsilon$ and $T + \epsilon$, and to shrink if it is out of this range. In this work, we employ a multi-feature image term $I_{tex}(p)$ [36] that takes into consideration the features (denoted as a set M) generated for a given volume resolution. For each surface point p, a subset of features S_p is defined as:

$$S_p = \left\{m \in M : |m(p) - \bar{x}(m)| > \theta \cdot \sigma(m)\right\}, \tag{4}$$

where m is a feature in set M, $\bar{x}(m)$ and $\sigma(m)$ are the feature mean and standard deviation inside the initial surface, $m(p)$ is the value of feature m in the point/voxel p and θ is a user-defined sensitivity parameter. In other words, the S_p set will contain all the features that diverge from their mean values inside the initial surface by θ standard deviations in voxel p; hence, the texture data term is defined as:

$$I_{tex}(p) = \begin{cases} v & |S_p| = 0 \\ -v & \text{otherwise,} \end{cases} \tag{5}$$

where v is a predefined constant (equal to 2 by default to balance the curvature influence and to ensure numerical stability). This term prompts the surface to expand into regions where the features are similar to the interior of the initial contour. When at least one feature is sufficiently different, the surface is prompted to retract.

B. LEVEL SET EVOLUTION

The proposed method employs two distinct approaches to the evolution of the surface at different resolutions. In the case of downscaled volumes, all the discrete coordinates of the ϕ function are updated. This reduces the chance of convergence to a local minimum, while the GPU-accelerated implementation provides acceptable performance. While working on the original volume resolution, the process switches to an optimised block-based sparse implementation that updates ϕ only in the neighbourhood of the second to last surface (i.e. around the zero-crossing of the level set function). Both methods are based on an upwind-like numerical scheme. The ϕ update at point p and iteration n is according to:

$$\phi^{n+1}(p) = \phi^n(p) + \Delta t F(p)|\nabla\phi(p)|, \tag{6}$$

where Δt is the time step and $|\nabla\phi(p)|$ is the length of ϕ gradient, calculated according to the front direction, based on the sign of F [22].

The block-based scheme is presented in Fig. 4. Firstly, the ϕ function from the last iteration is upscaled to the original resolution. Next, its volume is divided into blocks and a neighbourhood with a given voxel radius is calculated around the surface. The blocks with the voxels of this neighbourhood are then extracted and compacted into a reduced volume.

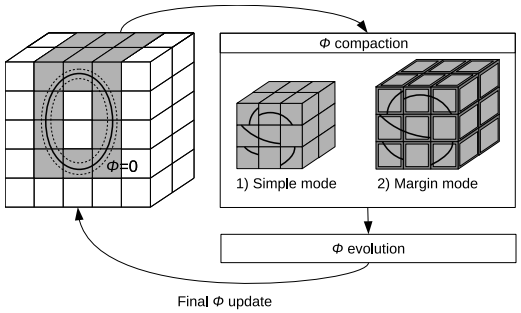

FIGURE 4. Compaction of the ϕ regions before the final stage of the evolution. The blocks containing the surface neighbourhood are packed into a smaller buffer that is sent to the GPU. After being processed, the buffer is unpacked and the final form of the ϕ function is updated.

Such a packed dataset is processed on the GPU. Since the compacted data contains only a fraction of the original volume, the computations and memory requirements are considerably decreased. Ultimately, the result is unpacked and used to update the ϕ function to its final form.

The proposed method provides two compaction modes with different versions of the speed function F. Both modes handle the ϕ curvature in a different way. The curvature requires multiple finite differences to be computed for each point, which causes an issue with the voxels on the borders of the blocks, due to them being separated from their original neighbourhood. The first mode is simplified, since it ignores the curvature term altogether and only the I_{tex} term is used. To prevent irregular surface borders, the level set function is post-processed after convergence and unpacking with a filter that performs a box blur on each of the ϕ points. The filter also slightly dilates the surface to compensate for eventual shrinkage.

The second mode performs the compaction with a margin of one voxel around each block. The curvature term is then included in the speed function but is computed only for the voxels inside the block and skipped for the margin. Given the voxel set B as the border of the block, the speed function $F_{block}(p)$ is defined as:

$$F_{block}(p) = \begin{cases} \alpha I_{tex}(p) & \text{if } p \in B \\ \alpha I_{tex}(p) + (1-\alpha)C(p) & \text{otherwise.} \end{cases} \quad (7)$$

This version gives a better approximation of the full-domain evolution at the cost of the increased size of the compacted data (due to additional border voxels).

C. IMPLEMENTATION

In this section, we describe the implementation regarding GPU utilisation and texture features.

1) FRAMEWORKS AND GPU UTILIZATION

The proposed implementation is based on OpenCL™ [37], a cross-platform framework for heterogeneous parallel programming, widely used for general-purpose computing on graphics processing units (GPGPU). OpenCL™ enables the utilisation of the GPU processing power, while reducing the necessity to fit the adapted algorithm into purely graphics-specific primitives. At its core, GPUs are designed for stream processing (i.e. for parallel application of a function (*kernel*) to each element of an input buffer, creating an output stream).

The method is implemented in MESA [38]—a platform for the design and evaluation of deformable model-based segmentation methods. The implementation uses Java for the main algorithm and the DWT. C language is used for the GPU acceleration in OpenCL™, which is applied in the level set evolution and GLCM features calculation. Both algorithms can process each voxel independently, which makes them suitable for running on a GPU. The level set evolution does not require a great deal of additional memory; for the GLCM features, however, some optimisations are necessary to conform to the specifics of the GPU.

2) TEXTURE FEATURES

At each step, the proposed method analyses the texture features generated for the given scale. In fact, any texture descriptor can be applied here, as long as it generates values for every voxel.

The presented implementation employs DWT- and GLCM-based features. These two algorithms work in notably different ways: the GLCM features are calculated for each separate resolution of the volume, which gives a vector of values for each volume point. The input 3D volume is passed to the GPU as a 1D buffer with values (voxels) stored in row–major order. The buffer is accessed only for discrete coordinates and no interpolation (which is possible with OpenCL™ data-types) is necessary. During the computations, a complete co-occurrence matrix has to be computed for the neighbourhood of each voxel. This can pose a problem for the GPU, since the private memory of a kernel is limited and dependent on the specific GPU architecture. The matrix is usually sparse; therefore, a quantisation scheme [9] (32 greyscale levels instead of the typical 256) is applied to reduce the memory consumption. This matrix is then used to calculate the final GLCM features. The final number of features depends on the combination of selected GLCM windows, orientations and voxel distances. In the current implementation, contrast, homogeneity and correlation are generated. The voxel step is set to 1 and the calculations are always performed for a 3^3 voxel cube since the scaling is performed at the volume level. The features can be calculated for 13 directions (see Fig. 1), or for all directions combined, which drastically reduces the search space at the cost of losing some directional information.

The DWT (performed using the Haar wavelet) is calculated at the beginning of the method. The initial 3D hyperbolic decomposition is performed with the JWave library,[1] which results in a coefficient volume equal in size to the input data. Next, the coefficients at a given decomposition level are selected for the specific stage of the method according to the resolution (the first level for the last stage, the second

[1] https://github.com/graetz23/JWave

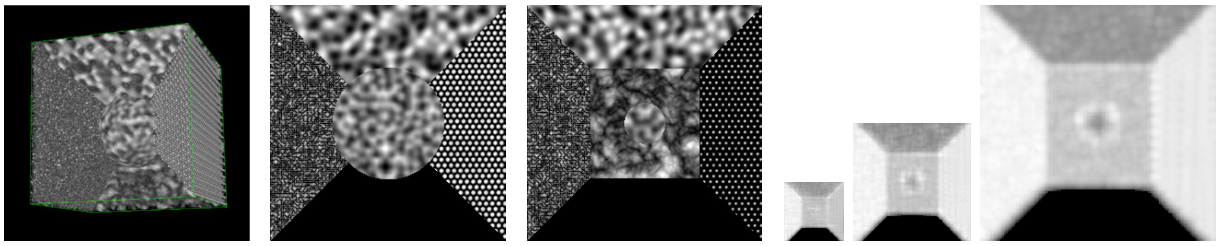

FIGURE 5. Synthetic texture mosaic volume: (a) 3D volume rendering, two cross-sections at (b) z = 149 and (c) z = 189, and (d) DWT features at z = 189.

level for the second to the last stage, etc.). Since the DWT coefficients are sparse in nature, they are smoothed with a Gaussian filter (with σ of 1 voxel) after upsampling to the current scale. Next, two approaches are possible: each separate coefficient set can be used in the level set evolution (seven in total for a given scale) or they can be combined according to the method proposed in [14]. In contrast to the GLCM, the initial DWT computation is a one-time operation and is performed as a standard CPU procedure. The Gaussian smoothing during the extraction of the DWT coefficients is actually more computationally intensive, but benefits from a multi-threaded implementation.

Similarly to the level set evolution, the texture feature generation process undergoes some optimisations during the last stage of the method (original volume resolution). Instead of sending a merged set of texture feature volumes to the GPU, a single volume containing the textural data term (defined in (5)) is pre-computed before the start of the evolution. This operation provides only limited performance gains but significantly reduces the GPU memory consumption (only one buffer instead of separate buffers for each feature). Since the last stage works only on a subset of the image domain, the feature generation can be limited to only the relevant regions inside the corresponding ϕ blocks. In the proposed implementation, this approach was applied to the GLCM features.

IV. EXPERIMENTAL VALIDATION

The proposed method was evaluated using synthetic datasets containing volumetric textures as well as real 3D medical images. During the experiments, the initial surfaces were manually placed inside the desired regions. The method was run with three scales (0.25, 0.5, and 1.0) of the input volume, up to 500 level set iterations at each scale and the balancing parameter α was set to 0.25. Unless otherwise specified, the combined feature volumes were used, i.e. one for the DWT coefficient and three for the GLCM features, which resulted in 12 separate feature volumes at all scales. The experiments were performed on a workstation with an Intel® Xeon® E5-1620v2 CPU, 16 GB RAM and Nvidia Titan Xp GPU. More detailed execution time analysis was performed with different GPUs. The proposed method was also compared to two state-of-the-art GPU-accelerated level set methods: interactive GPU active contours (IGAC) proposed by Willcocks *et al.* [34] and work-efficient GPU level set (WELS), developed by Roberts *et al.* [1].

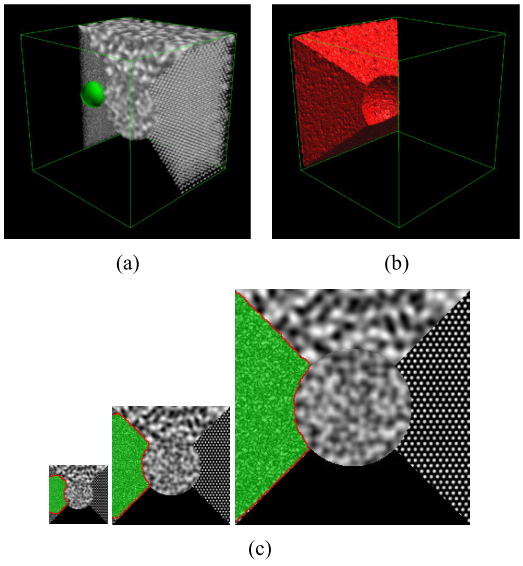

FIGURE 6. Segmentation result for the texture mosaic ($\theta = 4.0$): (a) sliced 3D rendering of the volume with the initial surface visible, (b) visualisation of the segmented surface, and (c) cross-sections of the segmented region after each stage of the process.

A. SYNTHETIC DATA SETS

Firstly, the method was tested on a synthetic 256^3 volume that contained a mosaic of five texture classes (see Fig. 5). The textures were created using a 3D noise generation library[2] and images from the Brodatz texture album. The results shown in Fig. 6 indicate that the obtained segmentation took about 7 s. The method could successfully differentiate between multiple regions with textures of different types and scales. Most of the target region was segmented during the first two stages (see Fig. 6(c)), which amounted to only 18% of the execution time. To match this result without the benefit of the multi-resolution approach, the algorithm required over 3200 level set iterations and needed to be run 2.5 times longer. The segmentation was also performed with the DWT features disabled (see Fig. 7). In this case, the surface exhibited a significant 'leakage' of the target region at the 0.5 resolution, but managed to correct itself in the final stage. Without the DWT-related calculations, the segmentation time was reduced to about 5.5 s.

The method was also tested using synthetic volumes from the RFAI database of 3D textures [39]. The database provides

[2]https://github.com/Auburns/FastNoise

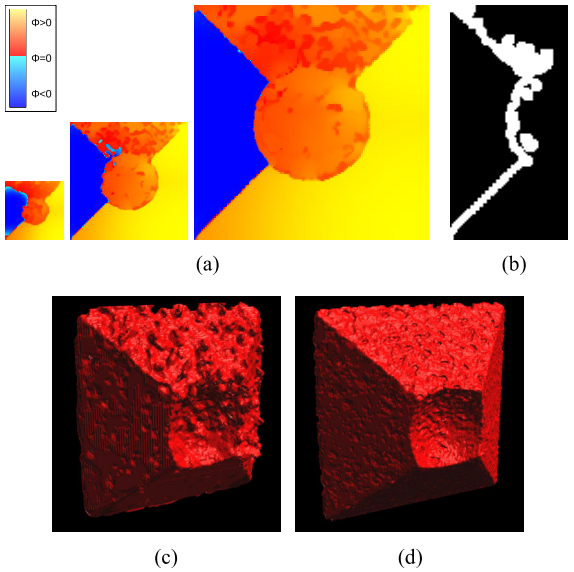

FIGURE 7. Segmentation of the texture mosaic ($\theta = 3.5$) with only GLCM features enabled: (a) slice of the ϕ function after each stage at $z = 128$, (b) the compaction region after the second stage, (c) the results after the second stage, and (d) the final surface.

multiple classes of 3D solid textures, as well as mosaics with the ground truth for segmentation evaluation. The datasets contain 128^3 volumes composed of two, three and four classes. The segmentation accuracy was assessed with four error measures: *volume overlap error* (VOE), *relative volume difference* (RVD), *Dice coefficient* (DC) and *average symmetric surface distance* (ASSD). Given two sets of voxels R and G, where R is the tested result and G is the ground truth segmentation, the first three measures were defined as:

$$\begin{aligned} VOE(G,R) &= 100\,(1 - |G \cap R|/|G \cup R|), \\ RVD(G,R) &= 100\,((|R| - |G|)/|G|), \\ DC(G,R) &= 2\,|G \cap R|/(|G| + |R|). \end{aligned} \tag{8}$$

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

$$ASSD(G,R) = \frac{1}{|S(G)| + |S(R)|}\left(\sum_{s_G \in S(G)} d(s_G, S(R)) + \sum_{s_R \in S(R)} d(s_R, S(G))\right) \tag{9}$$

The experiments were performed on five volumes composed of two classes (named *image1* to *image5* in the 2-class RFAI dataset). The average execution time was between 1 and 2 s. Fig. 8 shows the results for the first volume (*image1*). The method achieved high accuracy, in contrast to 3D implementations of a classical intensity-based level set method (ILS) [29], geometric active contour (GAC) [23], and ACWE [6]. The quality assessment is presented in Table 1 and Fig. 9 shows sample cross-sections with the segmented regions. The proposed method achieved DC equal to or exceeding 0.95 and VOE lower than 10 in all but one case. These values were quite good and the errors were located only on the borders of the regions. Although the voxel count of the border was low compared to the entire volume, the low values of ASSD were in line with the other quality measures. The first three datasets (*image1–3*) showed clearly acceptable results. The region border in *image3* (see Fig. 9(b)) was more irregular, but this could be improved by adjusting the α parameter to increase the influence of the curvature. The *image4* volume had a greater directional texture and required the calculation of the GLCM features for separate orientations (see Fig. 9(f)). The combined features were not sufficient to fully discriminate between the two regions and resulted in a significant under-segmentation: RVD $= -32.29$ and ASSD $= 5.58$, compared to final RVD $= -9.73$ and ASSD $= 1.48$ (see Fig. 9(e)). The last dataset (*image5*) was also challenging, since it contained two similar texture classes with difficult to distinguish borders, but the method managed to find most of the general outline of the region.

The ILS method used the image term presented in (3) and a GPU-based implementation. The T parameter was set to the mean intensity inside the initial surface, α was equal to 0.25 and only the ϵ tolerance was tuned. As expected, ILS was not well adapted to non-homogeneous regions. Its quality was relatively close to the proposed approach only for the first texture in *image1* (see Fig. 8(f)). In all other cases, the segmentation was largely unsuccessful (similar to Fig. 8(g)), with both high VOE and RVD and excessive ASSD. Furthermore, the ILS took more time to converge: about 11 s on average.

The GAC relied on an edge-based energy that pushed the surface towards the boundaries of the segmented region. This energy used a gradient of a smoothed version of the volume and diminished the closer the surface got to an edge. The smoothing was performed with a Gaussian filter with the σ parameter in a range from 2 to 4 pixels, which seemed appropriate for the dimension of the input volumes. The GAC approach was mostly inadequate for the RFAI dataset and usually resulted in significant undersegmentation, as indicated by high negative values of RVD that corresponded with VOE. The simple Gaussian filter was not able to simultaneously smooth the textures and preserve a sharp border between the patterns, therefore the surface was typically stopped before it could cover most of the segmented region.

Finally, the results of the proposed algorithm were compared with a 3D piecewise-constant version of the ACWE method. This approach partitions the dataset into two classes (phases) by minimizing their internal intensity variance. ACWE has an ability to deal with noise and the lack of clear borders between the regions, but works well only when the average intensity values of the phases are distinct enough.

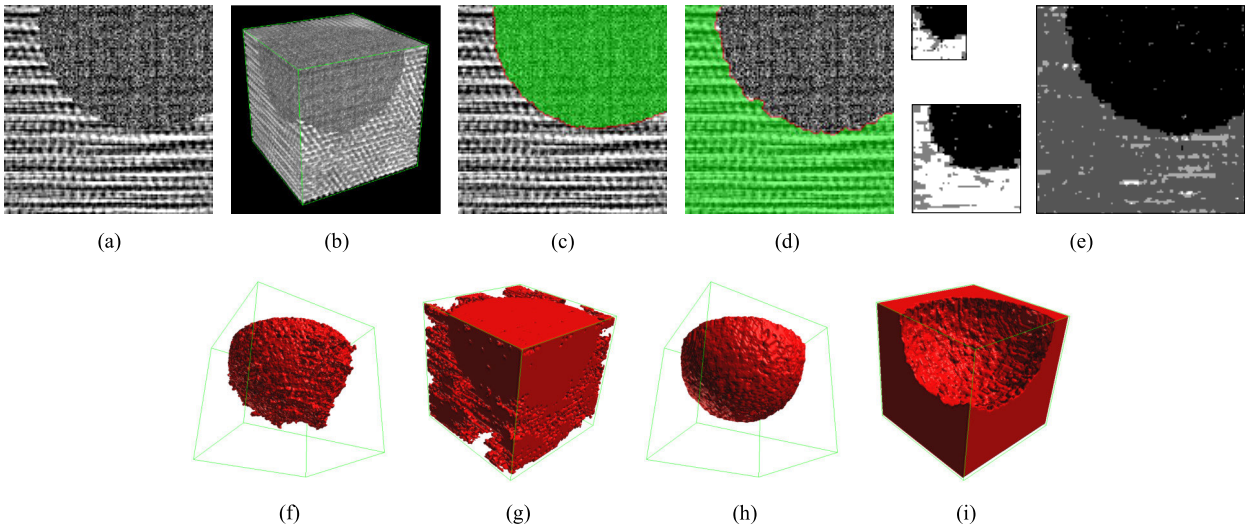

FIGURE 8. Segmentation result for the first RFAI 2-class volume (*image1*): (a) slice of the volume at z = 64, (b) 3D rendering of the volume, (c–d) cross-sections with the results of the proposed method (z = 64), (e) values of $|S_p|$ for the slice corresponding to z = 64 (mapped to greyscale), (f–g) 3D visualisations of the intensity-based level set results, and (h–i) 3D visualisations of the proposed method results.

TABLE 1. Accuracy of the RFAI 2-class dataset segmentation performed with the ILS, GAC, ACWE, and the proposed method (sensitivity parameters ϵ, σ, and θ also provided).

		ILS					GAC					ACWE					Proposed method				
Volume	Texture	ϵ	VOE	RVD	DC	ASSD	σ	VOE	RVD	DC	ASSD	VOE	RVD	DC	ASSD	θ	VOE	RVD	DC	ASSD
image1	class1	50	4.29	1.51	0.98	1.61	4	25.48	-25.48	0.85	10.53	1.86	1.34	0.99	0.60	2.00	2.17	1.57	0.99	0.70
	class2	65	65.07	-4.38	0.52	14.66	4	24.55	-24.55	0.86	17.27	0.87	-0.62	1.00	0.61	5.50	1.51	-0.84	0.99	0.89
image2	class1	38	50.12	96.73	0.67	11.17	2	20.87	-19.65	0.88	4.80	84.15	75.21	0.27	21.62	3.50	3.90	3.45	0.98	0.81
	class2	40	55.62	-55.12	0.61	26.40	4	95.61	-95.61	0.08	60.16	28.45	-11.74	0.83	21.43	6.00	0.62	-0.33	1.00	0.75
image3	class1	40	89.75	80.39	0.19	12.86	4	87.26	59.48	0.23	14.87	80.43	-36.38	0.33	11.57	4.00	6.90	-2.65	0.96	3.60
	class2	45	20.07	-1.07	0.89	13.35	4	24.03	-11.51	0.86	14.07	30.55	12.67	0.82	12.04	3.50	2.00	0.52	0.99	1.37
image4	class1	31	55.59	-41.93	0.62	6.58	2	80.15	82.42	0.33	13.65	100.0	119.44	0.00	42.03	6.00	9.73	-9.73	0.95	1.48
	class2	40	31.47	-24.57	0.81	27.01	4	23.78	-20.65	0.87	26.67	21.55	-8.64	0.88	41.50	3.50	0.54	0.47	1.00	1.23
image5	class1	37	57.74	-50.47	0.59	10.76	4	64.31	-63.63	0.53	9.76	53.17	-7.24	0.64	11.85	1.80	14.22	-5.30	0.92	4.02
	class2	39	42.13	-37.85	0.73	17.37	4	67.31	-41.58	0.49	15.55	24.43	2.93	0.86	13.52	1.85	7.62	-2.32	0.96	6.78

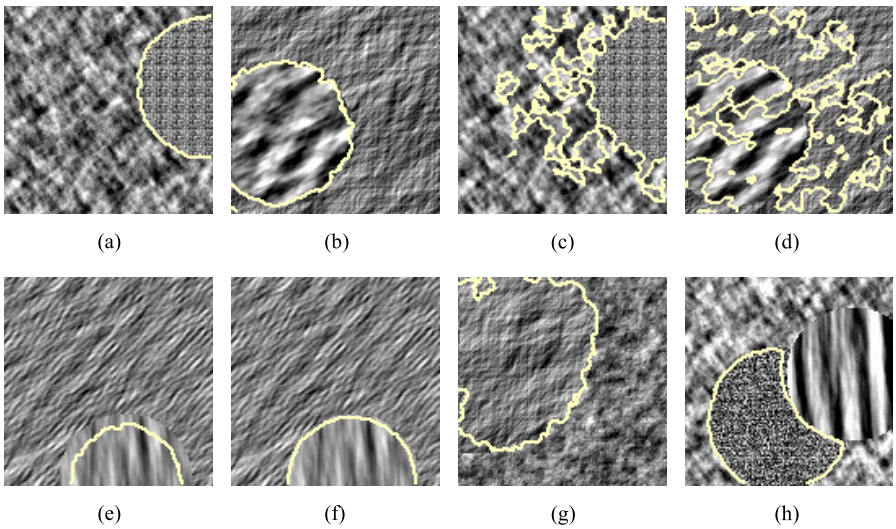

FIGURE 9. Example results for RFAI database volumes: (a) *image2* volume at z = 64, (b) *image3* at z = 64, (c–d) results of the intensity-based level on the two previous slices, (e) *image4* at z = 100 (undersegmentation of class1), (f) *image4* at z = 100 (final result of class1), (g) *image5* at z = 96 (class1), and (h) example result on a volume with multiple classes.

This condition was met for the *image1* volume, where the ACWE, unsurprisingly, achieved a high-quality segmentation. The other volumes, however, did not fit this criterion, as their textures differed more in scale, pattern, and orientation, rather than just in mean intensity. Ultimately, the ACWE segmentation of these volumes was largely unsuccessful.

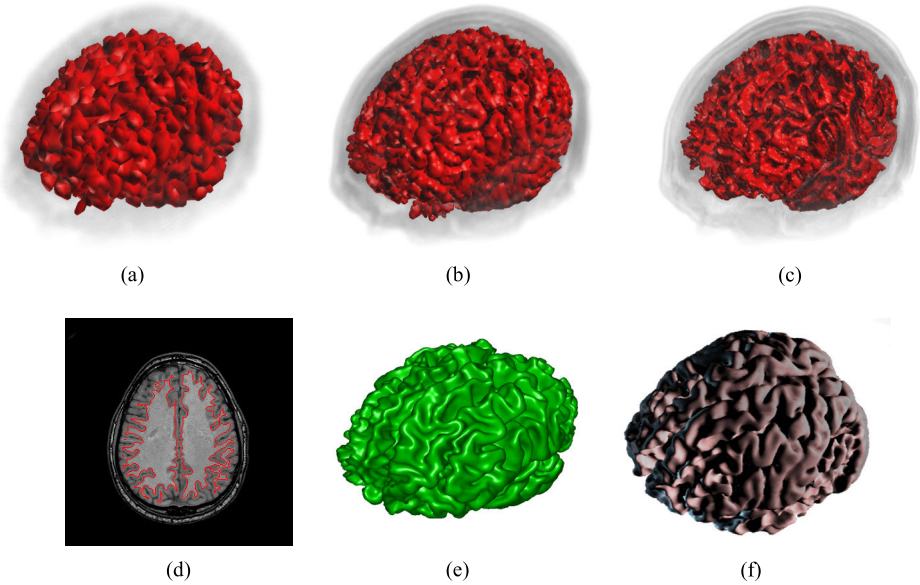

FIGURE 10. Segmentation of white matter in the brain MRI. Results of the proposed method at scales of (a) 0.25, (b) 0.5 and (c) 1.0, as well as (d) an example contour at z = 79. Surfaces in (a) and (b) are up-scaled for better presentation. Results of (e) WELS and (f) IGAC methods captured with their native software (the perceived perspective differences are due to the distinct rendering capabilities of the programs).

B. REAL DATA SETS

The proposed method was also tested using real 3D medical imaging data. The resulting quality and runtime performance was compared with the WELS and IGAC algorithms, since both of those methods are used for medical image segmentation. Although the accuracy of the outcomes is the ultimate goal for medical applications [40], [41], the complexity and time performance of the algorithms is also an important factor, especially considering the size of 3D imaging data [3].

The examples show the segmentation results of two volumes: a brain MRI provided in [1] (with 256 × 256× 174 voxels) and a thoracic CT scan from the 2017 Lung CT Segmentation Challenge (LCTSC) database [42]–[44] (256 × 256 × 168, cropped from 512^2 images for better scaling).

The segmentation of white matter in the brain dataset is presented in Fig. 10. The visualised surfaces show increasing segmentation fidelity at each scale. The method used the source data intensity and GLCM contrast as the image features and took about 3 s to complete. This performance was comparable with the WELS method (see Fig. 10(e)) which completed in a similar time. Regarding the IGAC (see Fig. 10(f)), the segmentation took between 20 and 35 s. Qualitatively, all three methods were able to obtain acceptable results, but runtime disparity was evident. The proposed approach was much faster than IGAC and on a par with WELS, despite the usage of more complex image feature descriptors.

The lung segmentation of the thoracic CT was performed according to the LCTSC guidelines (i.e. one lung should be separate from another and hilars and the trachea/main bronchus should be omitted). The results (see Fig. 11) show a successful segmentation with DC close to 0.98. The texture stopping term $|S_p|$ (see Fig. 11(d)) prevented the 'leakage' of the surface into the bronchus. Such behaviour is more difficult to achieve using only the image intensity, as observed for the segmentation using the WELS method (see Fig. 12). IGAC, due to its more advanced image term, is much more robust in segmenting only the lung region. The proposed method ran in 4.5 s, whereas WELS took 3 s and IGAC finished in about 40 s.

C. RUNTIME PERFORMANCE EVALUATION

The performance of the method was tested on three Nvidia GPUs: GTX 980, Titan Xp, and Tesla P100. The experiments were performed on the mosaic volume from the first example (256^3 voxels). Three default scales were used and level set iterations were fixed to 500 for each stage. Only the GLCM features were employed, since the DWT algorithm was implemented on the CPU. Table 2 shows the specifications of the cards and execution times for the final stage. Three modes were compared: an un-optimised version (Full), a sparse mode with a margin (Mrg.), and a sparse mode with smoothing and curvature disabled (NoC.). The execution times for the non-sparse version were included to illustrate the performance gains. The results showed that the sparse schemes significantly decreased the total time taken for segmentation. The combined time of the first two stages, independently of the last stage mode, took less than 20% of the total time in the un-optimised mode, indicating that most of the computations were performed in the final stage. Such disparities between the stages were expected, since each subsequent resolution increased the total voxel count eightfold.

Fig. 13 shows an in-depth analysis of the final stage. The stacked execution time of the separate GLCM and level

TABLE 2. Specifications of the graphics cards used for performance tests and total test volume segmentation times in different modes for the last resolution stage. A summary time of the first two stages is also provided.

Hardware specification				Total method time [s]			
Model	CUDA Cores	RAM [GB]	Clock [MHz]	Unoptimised	Sparse/Margin	Sparse/Smoothing	Stage 1+2 [s]
GTX 980	2048	4	1216	21.8	10.7	7.8	3.5
Titan Xp	3840	12	1582	10.2	6.2	5.1	2.0
Tesla P100	3584	16	1303	12.0	6.7	5.0	2.2

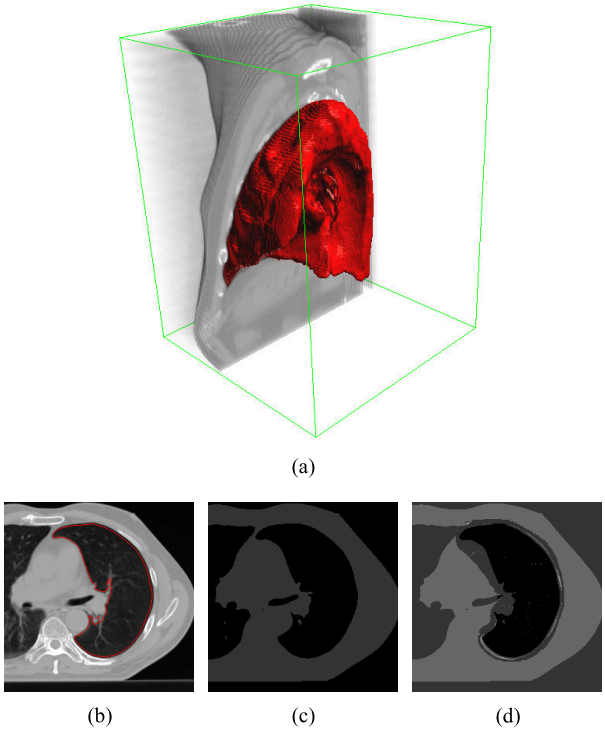

FIGURE 11. Lung CT segmentation using the proposed method: (a) sliced volume rendering of the data with the resulting surface visible, (b) resulting contour at z = 82, values of $|S_p|$ for the slice with (c) only the image intensity used and (d) with the GLCM features included.

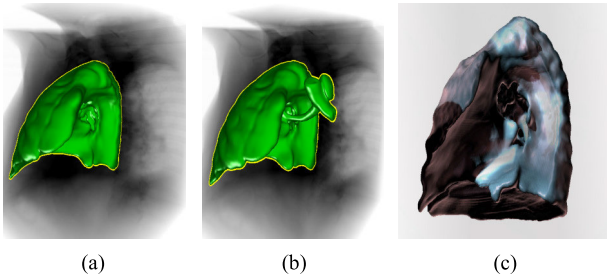

FIGURE 12. Lung CT segmentation with the (a–b) WELS and (c) IGAC methods. WELS was able to successfully segment the region (a), but precise parameter tuning was required to prevent leakage into the bronchus and trachea regions (b).

set (LS) stages are displayed for each GPU and final stage mode. Taking into account the capabilities of the GPUs, the results showed reasonable variation between the cards. The weakest GTX 980 gained the most from the optimisation schemes. Titan and Tesla differed in the GLCM and LS stages, but their overall time was very similar.

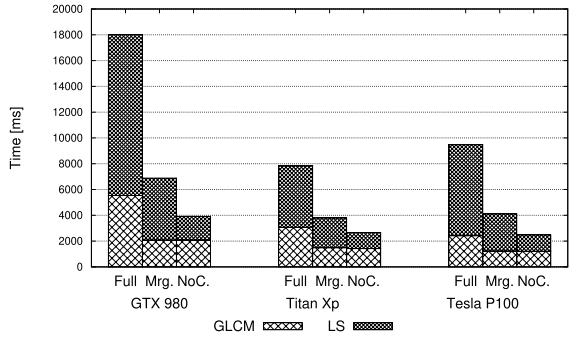

FIGURE 13. Performance of the last algorithm stage on different GPUs. Three modes are presented: unoptimised version (Full), sparse mode with margin (Mrg.) and sparse mode with smoothing and curvature disabled (NoC.). The stacked execution time consist of the level set evolution (LS) and GLCM generation.

Although the sparse mode with a margin gave results similar to the un-optimised mode, it took longer to complete than the simplified sparse mode with smoothing, because the margin mode had to perform the curvature computations. Furthermore, the additional border margin significantly increased the size of the compacted volume. With a block size of 8, an additional margin of 1 voxel almost doubled the packed data size (8^3 vs 10^3 voxels). In the test example, this resulted in a packed-to-original data ratio of 0.3 for the margin mode, compared to 0.16 for the smoothing scheme.

To put these results into perspective, the GPU-based 3D GLCM algorithm was compared to a single- and multi-threaded 2D CPU version. On an 8-core/16-thread AMD Ryzen^{TM} 2700 CPU, the generation of 2D features for each slice of the 256^3 volume took 201 s on a single thread and 39 s on all threads. Titan Xp could calculate more demanding 3D features of the same volume in 3.6 s, which provided $\times 59$ and $\times 10.8$ speedup over the CPU (using 4 GLCM orientations in 2D and 3D cases).

D. DISCUSSION

The proposed method demonstrated positive results for synthetic data sets and performed well on real volumetric images. The algorithm was able to use full 3D texture features while maintaining high performance, as supported by the comparison with other GPU-based level set implementations. Notably, WELS is still one of the fastest methods available. Its optimisations are based on maintaining a sparse narrow band around the evolving surface. The algorithm does not use tiling, but the band region is tracked in a per-voxel way, which provides an even bigger reduction in computational load. Its intensity-based image force, however, is similar to ILS and

is not well-suited for handing textured regions. Despite the costly generation of full 3D features, the proposed method is very competitive with WELS in terms of its pure execution time, obtaining segmentation results on real data sets in less than 5 s.

The IGAC uses a variational region-based scheme [6]. This approach evolves the level set functions according to the parameters of foreground and background regions. The image fitting term is global and has to be constantly updated during the evolution. Furthermore, the IGAC uses a Gaussian distribution-based image term that applies better to regions with non-uniform intensity. The work is therefore performed on the entire image domain and is bound to the Gaussian kernel calculation. In comparison to the proposed method, the IGAC was therefore significantly slower. However, the model used in IGAC has some advantages, such as lower dependence on initialisation—its global nature makes it harder to stuck in a local minimum. It has also been noted that the implementation focuses on interactive segmentation, whereby the operator can use brush-like tools to aid or block the surface progression. The algorithm parameters can also be adjusted in real time.

V. CONCLUSION AND FUTURE WORKS

In this article, a texture-based level set method for the segmentation of 3D volumes is proposed. The method takes advantage of a multi-resolution and sparse scheme to speed up computations and utilises the multi-scale nature of the provided texture features. The high performance is also enhanced by GPU acceleration using OpenCL™. The algorithm has the advantage of possibly using different texture features. The textural level set speed function utilises only the relevant features at a given scale and is not bound to specific descriptors. The experiments using synthetic datasets indicated that it can handle a large variety of textured volumes. Moreover, the performance evaluation of the method showed that it can efficiently process typical datasets, even on a current mid-range GPU, and does not require specialised hardware.

The promising test using real datasets suggested possible applications for medical image segmentation. 3D medical imaging datasets are traditionally presented as a series of 2D images and many previous methods were limited to slice-by-slice processing of the volumes [40], [45]. Given the size of modern imaging data, the segmentation of 3D structures on individual images can be a tedious and error-prone task. Full 3D algorithms [3], [41] offer significant advantages, since the continuity of the segmented region is more naturally maintained, without the need for post-processing of separate images. There is also no need for re-initialisation for separate slices. The presented examples show that the method is able to successfully segment some typical medical datasets in a matter of seconds, while the texture-based approach can be beneficial for the quality of the results.

The current version of the method could be improved in many ways. The textural level set speed function has an ability to discard irrelevant features at a given moment. The method, however, has to calculate the features anyway before starting the evolution. A pre-selection scheme could be applied here to limit those computations (e.g. by trying to predict the most useful features, using data from the earlier stages and scales). Integration with region-based models [6] could also be an interesting direction to take. Currently, the method relies on manual initialisation, which could also be improved by using region-based and global minimisation schemes [31].

Regarding the performance, the employed optimisation relies on limiting the computations to the locally relevant data regions. To effectively run for a typical uniformly distributed GPU workload, this kind of approach requires relatively complex packing and streaming schemes in order to achieve an optimal GPU saturation. This issue could be addressed by the utilisation of more advanced GPGPU computing models, such as OpenCL™ non-uniform workgroups or CUDA dynamic parallelism [46], which are designed for irregular workloads. From an implementation standpoint, the usage of private/local GPU memory for GLCM and level set algorithms could be improved to better accommodate the capabilities of modern hardware.

REFERENCES

[1] M. Roberts, J. Packer, M. Sousa, and J. Mitchell, "A work-efficient GPU algorithm for level set segmentation," in *Proc. Conf. High Perform. Graph. (HPG)*, 2010, pp. 123–132.

[2] J. Roels, J. De Vylder, Y. Saeys, B. Goossens, and W. Philips, "Decreasing time consumption of microscopy image segmentation through parallel processing on the GPU," in *Proc. Int. Conf. Adv. Concepts Intell. Vis. Syst. (ACIVS)*. Cham, Switzerland: Springer, 2016, pp. 147–159.

[3] E. Smistad, T. L. Falch, M. Bozorgi, A. C. Elster, and F. Lindseth, "Medical image segmentation on GPUs—A comprehensive review," *Med. Image Anal.*, vol. 20, no. 1, pp. 1–18, Feb. 2015.

[4] P. Moore and D. Molloy, "A survey of computer-based deformable models," in *Proc. Int. Mach. Vis. Image Process. Conf. (IMVIP)*, Sep. 2007, pp. 55–66.

[5] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, Jan. 1988.

[6] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 266–277, Feb. 2001.

[7] R. Ronfard, "Region-based strategies for active contour models," *Int. J. Comput. Vis.*, vol. 13, no. 2, pp. 229–251, Oct. 1994.

[8] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973.

[9] L. Tesař, A. Shimizu, D. Smutek, H. Kobatake, and S. Nawano, "Medical image analysis of 3D CT images based on extension of haralick texture features," *Comput. Med. Imag. Graph.*, vol. 32, no. 6, pp. 513–520, Sep. 2008.

[10] J. J. Cerrolaza, N. Safdar, C. A. Peters, E. Myers, J. Jago, and M. G. Linguraru, "Segmentation of kidney in 3D-ultrasound images using Gabor-based appearance models," in *Proc. IEEE 11th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2014, pp. 633–636.

[11] M. Gao, H. Chen, S. Zheng, and B. Fang, "A factorization based active contour model for texture segmentation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 4309–4313.

[12] Y. Dong, H. Zhang, Z. Liu, C. Yang, G.-S. Xie, L. Zheng, and L. Wang, "Neutrosophic set transformation matrix factorization based active contours for color texture segmentation," *IEEE Access*, vol. 7, pp. 93887–93897, 2019.

[13] Y. D. Cid, H. Muller, A. Platon, P.-A. Poletti, and A. Depeursinge, "3D solid texture classification using locally-oriented wavelet transforms," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1899–1910, Apr. 2017.

[14] A. Achuthan, M. Rajeswari, D. Ramachandram, M. E. Aziz, and I. L. Shuaib, "Wavelet energy-guided level set-based active contour: A segmentation method to segment highly similar regions," *Comput. Biol. Med.*, vol. 40, no. 7, pp. 608–620, Jul. 2010.
[15] O. Pujol and P. Radeva, "Texture segmentation by statistical deformable models," *Int. J. Image Graph.*, vol. 4, no. 3, pp. 433–452, Jul. 2004.
[16] X. Huang, Z. Qian, R. Huang, and D. Metaxas, "Deformable-model based textured object segmentation," in *Proc. Int. Workshop Energy Minimization Methods Comput. Vis. Pattern Recog. (EMMCVPR)*. Berlin, Germany: Springer, 2005, pp. 119–135.
[17] Q. Wu, Y. Gan, B. Lin, Q. Zhang, and H. Chang, "An active contour model based on fused texture features for image segmentation," *Neurocomputing*, vol. 151, pp. 1133–1141, Mar. 2015.
[18] S. Luo, L. Tong, and Y. Chen, "A multi-region segmentation method for SAR images based on the multi-texture model with level sets," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2560–2574, May 2018.
[19] F. S. Al-Qunaieer, H. R. Tizhoosh, and S. Rahnamayan, "Multi-resolution level set image segmentation using wavelets," in *Proc. 18th IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2011, pp. 269–272.
[20] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 674–693, Jul. 1989.
[21] S. G. Roux, M. Clausel, B. Vedel, S. Jaffard, and P. Abry, "Self-similar anisotropic texture analysis: The hyperbolic wavelet transform contribution," *IEEE Trans. Image Process.*, vol. 22, no. 11, pp. 4353–4363, Nov. 2013.
[22] J. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge, U.K.: Cambridge Univ. Press, 1999.
[23] V. Caselles, F. Catté, T. Coll, and F. Dibos, "A geometric model for active contours in image processing," *Numer. Math.*, vol. 66, no. 1, pp. 1–31, Dec. 1993.
[24] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *Int. J. Comput. Vis.*, vol. 22, no. 1, pp. 61–79, 1997.
[25] D. Mumford and J. Shah, "Optimal approximations by piecewise smooth functions and associated variational problems," *Commun. Pure Appl. Math.*, vol. 42, no. 5, pp. 577–685, Jul. 1989.
[26] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, "A PDE-based fast local level set method," *J. Comput. Phys.*, vol. 155, no. 2, pp. 410–438, Nov. 1999.
[27] B. Houston, M. B. Nielsen, C. Batty, O. Nilsson, and K. Museth, "Hierarchical RLE level set: A compact and versatile deformable surface representation," *ACM Trans. Graph.*, vol. 25, no. 1, pp. 151–175, Jan. 2006.
[28] M. B. Nielsen and K. Museth, "Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets," *J. Sci. Comput.*, vol. 26, no. 3, pp. 261–299, Mar. 2006.
[29] A. Lefohn, J. Cates, and R. Whitaker, "Interactive, GPU-based level sets for 3D segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent. (MICCAI)*. Berlin, Germany: Springer, 2003, pp. 564–572.
[30] C. Li, C. Xu, C. Gui, and M. D. Fox, "Distance regularized level set evolution and its application to image segmentation," *IEEE Trans. Image Process.*, vol. 19, no. 12, pp. 3243–3254, Dec. 2010.
[31] X. Bresson, S. Esedoḡlu, P. Vandergheynst, J.-P. Thiran, and S. Osher, "Fast global minimization of the active contour/snake model," *J. Math. Imag. Vis.*, vol. 28, no. 2, pp. 151–167, Aug. 2007.
[32] M. Rumpf and R. Strzodka, "Level set segmentation in graphics hardware," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, vol. 3, Oct. 2001, pp. 1103–1106.
[33] A. C. Jalba, W. J. van der Laan, and J. B. T. M. Roerdink, "Fast sparse level sets on graphics hardware," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 1, pp. 30–44, Jan. 2013.
[34] C. G. Willcocks, P. T. G. Jackson, C. J. Nelson, A. V. Nasrulloh, and B. Obara, "Interactive GPU active contours for segmenting inhomogeneous objects," *J. Real-Time Image Process.*, vol. 16, no. 6, pp. 2305–2318, Dec. 2019.
[35] L. Wang, L. He, A. Mishra, and C. Li, "Active contours driven by local Gaussian distribution fitting energy," *Signal Process.*, vol. 89, no. 12, pp. 2435–2447, Dec. 2009.
[36] D. Reska, C. Boldak, and M. Kretowski, "Toward texture-based 3D level set image segmentation," in *Image Processing and Communications Challenges 7* (Advances in Intelligent Systems and Computing). Cham, Switzerland: Springer, 2016, pp. 205–211.
[37] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Comput. Sci. Eng.*, vol. 12, no. 3, p. 66, 2010.
[38] D. Reska, K. Jurczuk, C. Boldak, and M. Kretowski, "MESA: Complete approach for design and evaluation of segmentation methods using real and simulated tomographic images," *Biocybern. Biomed. Eng.*, vol. 34, no. 3, pp. 146–158, 2014.
[39] L. Paulhac, P. Makris, and J.-Y. Ramel, "A solid texture database for segmentation and classification experiments," in *Proc. Int. Conf. Comput. Vis. Theory Appl (VISSAPP)*, 2009, pp. 135–141.
[40] A. Mansoor, U. Bagci, B. Foster, Z. Xu, G. Z. Papadakis, L. R. Folio, J. K. Udupa, and D. J. Mollura, "Segmentation and image analysis of abnormal lungs at CT: Current approaches, challenges, and future trends," *Radiographics*, vol. 35, no. 4, pp. 1056–1076, Jul. 2015.
[41] L. E. Carvalho, A. C. Sobieranski, and A. von Wangenheim, "3D segmentation algorithms for computerized tomographic imaging: A systematic literature review," *J. Digit. Imag.*, vol. 31, no. 6, pp. 799–850, Dec. 2018.
[42] J. Yang, G. Sharp, H. Veeraraghavan, W. van Elmpt, A. Dekker, T. Lustberg, and M. Gooding, "Data from lung CT segmentation challenge," *Cancer Imag. Arch.*, 2013, doi: 10.7937/K9/TCIA.2017.3r3fvz08.
[43] J. Yang, H. Veeraraghavan, S. G. Armato, K. Farahani, J. S. Kirby, J. Kalpathy-Kramer, W. van Elmpt, A. Dekker, X. Han, X. Feng, P. Aljabar, B. Oliveira, B. van der Heyden, L. Zamdborg, D. Lam, M. Gooding, and G. C. Sharp, "Autosegmentation for thoracic radiation treatment planning: A grand challenge at AAPM 2017," *Med. Phys.*, vol. 45, no. 10, pp. 4568–4581, Oct. 2018.
[44] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle, L. Tarbox, and F. Prior, "The cancer imaging archive (TCIA): Maintaining and operating a public information repository," *J. Digit. Imag.*, vol. 26, no. 6, pp. 1045–1057, Dec. 2013.
[45] T. Ivanovska, K. Hegenscheid, R. Laqua, S. Gläser, R. Ewert, and H. Völzke, "Lung segmentation of MR images: A review," in *Visualization in Medicine and Life Sciences III*. Cham, Switzerland: Springer, 2016, pp. 3–24.
[46] J. Wang and S. Yalamanchili, "Characterization and analysis of dynamic parallelism in unstructured GPU applications," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Oct. 2014, pp. 51–60.

DANIEL RESKA received the M.S. degree from the Bialystok University of Technology, Poland, in 2010, where he is currently pursuing the Ph.D. degree with the Faculty of Computer Science. His research interests focus on image segmentation, medical image analysis, and parallel computing.

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