Efficient Representation of Geometric Tree Models with Level-of-Detail Using Compressed 3D Chain Code

Damjan Strnad, Štefan Kohek, Andrej Nerat, and Borut Žalik

Abstract—In the paper, we present a method for space-efficient representation of geometric tree models, which are provided as skeletons with radii attached to individual branch segments. The proposed approach uses a new differential 3D chain code to encode orientation changes of consecutive branch segments, which allows optimizing chain code generation for increased compressibility while maintaining control over the model reconstruction error. The presented method is the first to encode the complete branching geometry including the branch radii and provides level-of-detail construction directly from the chain code. It is demonstrated that by using interpolative encoding of the resulting tree descriptors and radii sequences, the storage requirements for geometric description of a mixed all-aged forest can be reduced to less than 15 percent of its raw size while preserving the structural fidelity of tree models.

Index Terms—Geometric tree model, differential chain code, interpolative compression, level-of-detail

1 Introduction

Today, there is an increasing demand for detailed monitoring and modeling of surroundings [1], [2], where trees represent an important variable element of the natural environment. Building geometric models of trees to a suitable level of structural accuracy is important for both analytical and predictive purposes [3], [4], [5], [6], [7]. Space-efficient representation of trees is a challenging task due to the high complexity and variability of their ramification patterns. The main sources of geometric tree models are reconstruction from remotely sensed data [8], [9], [10], [11], [12], [13], [14], [15], [16], tree growth simulation [17], [18], and data-driven statistical modeling of trees [19]. When these methods are applied on extensive areas, the number of resulting geometric models can reach millions, which makes their efficient storage highly desirable. In order to improve visualization rates, full resolution tree models are often down-sampled into multiple level-of-detail (LOD) representations [20]. These reduced representations need to be derived on-the-fly or explicitly stored alongside the originals, which emphasizes the need for efficient storage of tree models.

In this paper, we introduce a new method for space-efficient encoding of geometric tree models. The branching structure of a model is given as a tree skeleton, with branch radii assigned to individual linear segments called internodes. In the proposed approach, a 3D chain code is used to describe the orientations of individual internodes. When the tree skeleton is traversed during encoding, the main branch extension is followed preferentially at each fork, which produces smoother transition of both internode orientations and radii. By using a new chain code that encodes local orientation changes from one internode to the next, the resulting descriptor length is reduced substantially. The method allows setting the upper bound of reconstruction error through resolution adjustment, and supports the derivation of LOD hierarchy directly from the chain code sequence of the full-detail model. The chain code string and the sequence of corresponding radii are compressed using an adaptation of the lossless interpolative encoding scheme, which results in an efficient reduction of tree models' storage size.

The problem addressed in this paper and its proposed solution are related to the general issue of discrete 3D curve representations, for which extensions of 2D chain code-based solutions have been developed [21], [22]. Branch encoding support was added to describe skeletonized objects [23] and enclosing trees of voxel-based objects [24], [25]. A more specific application of 3D chain codes for representation of tree objects was introduced by Bribiesca [26] and used in tree symmetry detection [27]. The idea was later extended by Sánchez-Cruz et al. [28], who proposed a 25-symbol relative chain code for encoding 3D skeletons.

Although conceptually related, our approach differs considerably from these studies by targeting tree models represented as skeletons of connected branch segments with corresponding radii. Instead of tracing the digitalized model structure by encoding 3D paths voxel by voxel, the proposed 3D chain code encodes differences in end-to-end orientations of successive internodes. We therefore call such an encoding mechanism a *differential chain code*. Previous approaches also do not deal specifically with tree descriptor LODs or compression. In this paper, both of these issues are considered, because they are crucial for minimizing the required storage

• *The authors are with the University of Maribor, Maribor 2000, Slovenia. E-mail: {damjan.strnad, stefan.kohek, andrej.nerat, borut.zalik}@um.si.*

Manuscript received 27 Sept. 2018; revised 9 May 2019; accepted 14 June 2019. Date of publication 24 June 2019; date of current version 6 Oct. 2020. (Corresponding author: Damjan Strnad.)
Recommended for acceptance by S. Schaefer.
Digital Object Identifier no. 10.1109/TVCG.2019.2924430

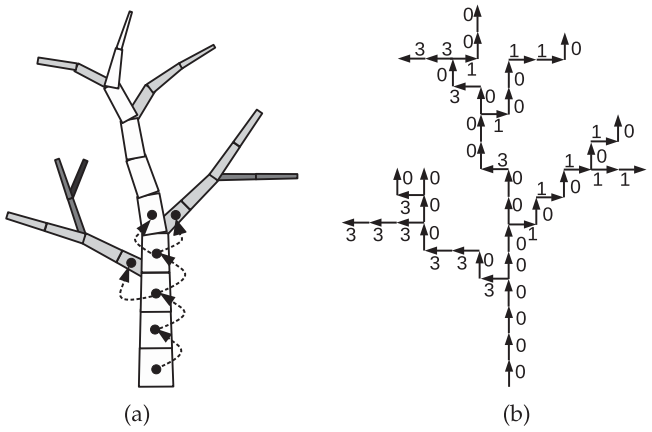

Fig. 1. Tree ramification hierarchy as (a) a chain of pointer-connected branch segments called internodes (the darker shade denotes the increasing branch level), and (b) a chain coded sequence of axis-aligned movements along the tree skeleton.

space for large amounts of tree geometry that needs to be retrieved and visualized frequently, e.g., in forest development analysis and simulation [29].

The main contributions of this paper are:

- a differential 3D chain code for encoding the orientation changes in internode sequences,
- an LOD scheme for derivation of geometrically reduced tree models directly from their chain code representation,
- a single interpolative compression mechanism for complete branching geometry, including the branch radii.

The organization of the paper is as follows. In Section 2, the background on techniques that form the basis of the proposed tree encoding method is provided, while the method itself is described in Section 3. The experimental results are reported in Section 4 and the limitations of the method are discussed in Section 5. The paper concludes with a summary in Section 6.

2 Background

The goal of this study is to provide a solution for reducing the off-line storage requirements of geometric tree models, as well as their memory footprint when they are loaded for rendering or processing. The storage reduction is achieved by two complementary mechanisms–chain code-based representation and lossless compression of the resulting descriptors. The third mechanism is tree model simplification by means of LOD reduction. While the chain coding and compression retain the underlying structural complexity of an object, the LOD reduction simplifies the object by transforming its geometric representation to a lower level of detail.

In the process of model representation change, some amount of accuracy loss is acceptable, especially for trees that have been reconstructed from noisy remote sensing data, generated by a simulation, or randomly sampled from a statistical model [19]. To model branches, we use generalized cylinders approximated by chains of geometric primitives, which is a popular practice in computer graphics [30], [31], [32], [33], [34], [35], [36]. The complete tree skeleton is a topologically connected set of branch chains, which consist of fixed-length internodes with defined top and bottom thickness (Fig. 1a). In geometric modeling of trees, branch thickness is often determined by some countable branch property [37], [38], [39], [40]. During visualization, such integral values are converted into actual branch radii using the rules that are based on botanical observations [37], [41].

In the following subsections, we provide a short overview of the three techniques involved in storage reduction for 3D geometric models of trees. Their incorporation in the proposed chain code-based solution is presented in the next section.

2.1 Chain Code Representation of 3D Ramified Objects

Chain codes have been developed as an efficient way to describe rasterized 2D shapes, where a small alphabet of symbols is used to encode movements along the shape's border or skeleton [42], [43], [44], [45]. Each symbol in a chain code sequence describes a move to the neighboring cell in the corresponding direction (Fig. 1b). The chain code is called absolute if the same symbol always denotes movement in the same Cartesian direction, and relative if a symbol denotes a change of direction with respect to previous movement. The advantage of relative coding is that the descriptors are invariant under rotation, which allows comparison of shapes. In cases where this is not important, the use of an absolute chain code offers simpler interpretation. Extensions of chain codes into 3D, where the chain code sequence traces a voxelized path, have been proposed for the description of 3D curves [22] or enclosing surfaces [46]. For any chain code, there is a trade-off between the size of the alphabet and the length of resulting shape descriptors. By limiting the number of symbols, a 3D chain code can restrict the movements to face-connected, edge-connected, or vertex-connected neighbor voxels.

In order to describe ramified objects, the chain code's alphabet is extended with additional symbols that denote forking points (also called junctions) [26]. When hitting a forking point during path tracing, a separate chain code is produced for each of the path extensions from that fork. Arbitrary branching structures can then be described using nested chain codes for segments between consecutive forks.

A 3D chain code is only used to describe the positions and orientations of tree skeleton segments. Additional internode data (e.g., their thicknesses or radii) need to be stored in the same visiting order during the chain code generation, such that a sequential correspondence exists between the codes and geometric properties of internodes.

2.2 Level-of-Detail

LOD extraction is a widely studied problem in computer graphics and geometric modeling [47]. It refers to the reduction of 3D objects' geometry in order to support faster visualization [48]. In this study, we address the issue of LOD reduction for the encoded tree skeletons, while the related problem of foliage simplification can be addressed using existing solutions [33], [40], [49], [50], [51]. Geometrically impoverished models are derived from a full-resolution reference at multiple levels of decreasing detail in order to support smooth visual transitions for scene walkthroughs and

animations. Specific LOD solutions for trees have been proposed that simplify or prune parts of the model with intensity proportional to the distance from the camera [52], [53], [54], [55]. However, these approaches operate in the geometric domain, while, for a chain code-based representation of tree models, it is desirable to support direct extraction of the required LOD in order to avoid wasteful generation of geometry that is discarded later.

Volume partitioning methods have been proposed for simplifying 2D chain code-based descriptors in order to produce shape hierarchies with increasingly coarse detail [56], [57], but there are currently no reports about research on similar simplification of 3D codes. In addition, volume partitioning is not efficient for objects with sparse and spatially scattered geometry, such as trees. In the next section, an LOD mechanism for chain code-based tree descriptors is presented that prunes the underlying geometrical model based on a branch thickness threshold.

2.3 Interpolative Encoding

The space requirements for storing the chain coded tree models can be substantially reduced by compressing the raw chain codes and radii sequences. To this end, we apply the lossless interpolative compression mechanism on top of the chain coding of tree skeletons.

Interpolative encoding has been introduced as an efficient method for compression of inverted indexes in information retrieval systems [58]. An inverted index is an associative array that maps a vocabulary term to a sorted list of identifiers of documents in which the term appears. Owing to the temporal locality and topical similarity of processed documents, clusters of sequential IDs tend to form in the sorted list. This results in long sequences of small d-gaps, i.e., differences between consecutive values, which can be compressed effectively using interpolative encoding. In the next section, we exploit the fact that the sequences of internode radii in geometric tree model descriptions exhibit similar clustering of values as inverted indexes and can, therefore, be encoded efficiently by a properly adapted interpolative scheme. Moreover, since state-of-the-art results have been reported recently for interpolative compression of chain code-based descriptors of 2D shapes [59], we apply the same method to encode both the radii and the chain code sequences.

In the interpolative encoding, a non-decreasing integer sequence is subdivided recursively, and the middle element value is encoded with respect to the range of its possible values given the values of the first and the last subsequence elements. As an example, assume that the first and the last element values in a sorted sequence $Q[1] \dots Q[45]$ are 18 and 106, respectively. The value of the element at the middle position $m = 23$ is bounded to the range [18, 106], which can be encoded using $\lceil \log_2(106 - 18 + 1) \rceil = 7$ bits. The process is then repeated for the left subsequence $Q[1] \dots Q[23]$, whose middle element $Q[12]$ can be encoded using $\lceil \log_2(Q[23] - 18 + 1) \rceil$ bits, and for the right subsequence $Q[23] \dots Q[45]$, where the number of necessary bits for encoding the central element $Q[34]$ is $\lceil \log_2(106 - Q[23] + 1) \rceil$. If the maximal possible d-gap is known, which is the case with the sequences produced from the chain code symbols, even tighter bounds for the range of possible values for $Q[m]$ can be determined at each step. By using shorter codes for the values near the range center, as in e.g., FELICS encoding [60], the expected code length can be further reduced. In the process of decoding, the left and the right subsequence are processed recursively once the middle element value has been established.

3 The Proposed Tree Encoding Method

In this section, the main components of the proposed chain code-based encoding method for geometric tree models are described. The 3D chain coding of tree skeletons is presented first, followed by a description of an LOD reduction procedure that works directly with the underlying tree model by shortening its code. Finally, the necessary adaptations for the application of interpolative compression to the resulting chain code and radii sequences are explained.

3.1 Tree Skeleton Encoding

The input to the encoding procedure is a tree skeleton, which is an ordered sequence of tree segments called internodes. Each internode is described by 3D locations of its endpoints, the top and the bottom radius, and the pointers to the internode's successors in the tree structure. The encoding of a tree skeleton is performed by traversing it from the root in a depth-first manner and encoding the visited internodes sequentially. We propose a new 3D chain code for compact representation of the skeleton structure, which is differential in the sense that only the changes of internode end-to-end orientations are encoded. Because the starting point for the skeleton encoding is fixed to be the tree root and the skeleton geometry is expressed in a coordinate system with known orientation, the use of rotation invariant relative chain code is not necessary. The proposed differential mechanism is, thus, applied on top of a 6-connectivity 3D version of absolute Freeman chain code [42], which is used as the base code for encoding the orientations of tree skeleton elements. According to the established nomenclature, we shall refer to this code as F6. The code uses numeric characters *0-5* to represent the orthogonal moves of fixed length in either direction of three coordinate axes (Fig. 2a). In order to allow the encoding of branching and segmentation that are present in geometric tree models, the F6 alphabet is extended with three additional symbols. Besides the move symbols, the extended code uses character *6* to denote fork points in the skeleton, and character *7* to mark an end of a branch (Fig. 2b). Ending a branch subsumes the process of "deforking" to the next front forking position in a queue. Finally, because each internode orientation is encoded using multiple move symbols, character *8* is used to separate internodes in the code sequence. Having a special symbol for this purpose is necessary in order to define the differential code, which is described later. Whenever an internode is followed by a fork or a branch terminus, the use of symbol *8* for signaling internode endpoint is not required, because the respectively used characters *6* and *7* convey that information. The summary of symbol significances is given in Table 1.

In order to encode an internode orientation, one could apply the standard approach by tracing the voxelized path from one internode endpoint to another. However, since an internode is a linear element, the voxel coordinate difference between its endpoints carries all necessary information for reconstructing the internode orientation. Because every permutation of an absolute chain code describes the same

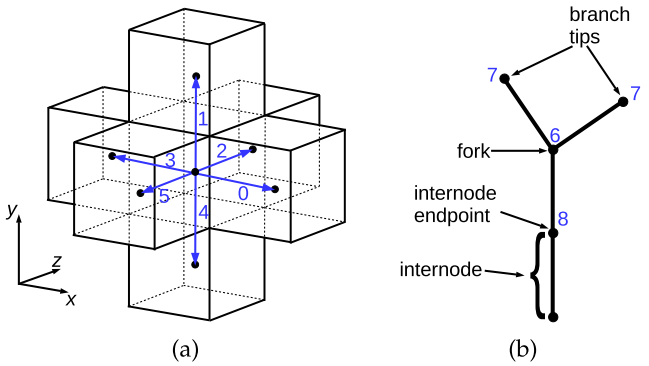

Fig. 2. The meaning of (a) movement symbols and (b) branching and segmentation symbols used by the extended F6 chain code.

internode orientation, we can choose the one with the best expected compression efficiency. With interpolative compression in mind, the best option is to cluster the symbols corresponding to the moves in the same direction, such that, for instance, the path *0101021* is actually stored as *0001112*.

The encoding of a tree skeleton always starts at the root internode, which is well defined for geometric tree models and starts at the origin of the tree's local coordinate system (CS). To encode the orientation of an internode with endpoints $\mathbf{p}_i = (p_{i,x}, p_{i,y}, p_{i,z})$ and $\mathbf{p}_j = (p_{j,x}, p_{j,y}, p_{j,z})$ as a sequence of fixed-length moves along the axes of the local CS, implicit voxelization is performed, in which the integer voxel coordinates for both internode endpoints are determined using:

$$\begin{aligned} \mathbf{v}_i &= \mathrm{rtz}\left(\frac{1}{d}\mathbf{p}_i + \frac{1}{2}\mathrm{sgn}(\mathbf{p}_i)\right) \\ \mathbf{v}_j &= \mathrm{rtz}\left(\frac{1}{d}\mathbf{p}_j + \frac{1}{2}\mathrm{sgn}(\mathbf{p}_j)\right). \end{aligned} \tag{1}$$

Here, rtz denotes the element-wise operation of rounding towards zero (i.e., truncating), and sgn is the sign function. Equation (1) aligns the origin of the tree's local CS with the center of voxel (0,0,0). In Equation (1), d is the voxel size set through the user-defined resolution r as:

$$d = \frac{l}{r}, \tag{2}$$

where l is the geometric internode length used for the construction of a geometric tree model. The resolution parameter r determines the number of voxels spanned by an axis-aligned internode and allows balancing between the accuracy

TABLE 1
Definition of Chain Code Symbols for Tree Model Encoding

Symbol	Meaning
0	move in direction $+x$
1	move in direction $+y$
2	move in direction $+z$
3	move in direction $-x$
4	move in direction $-y$
5	move in direction $-z$
6	forking point
7	branch tip
8	end of internode

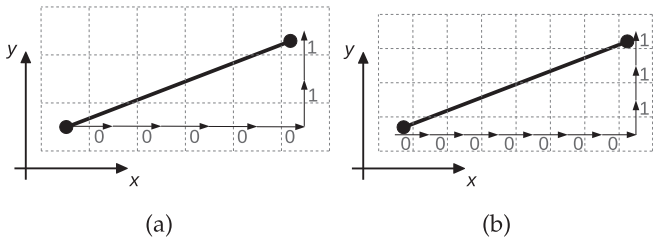

Fig. 3. A 2D example of end-to-end internode orientation encoding at resolutions (a) $r = 5$ and (b) $r = 7$, which produces respective descriptors *0000011* and *0000000111*.

of representation and the length of the obtained chain code strings (Fig. 3). The difference $\Delta\mathbf{v} = \mathbf{v}_j - \mathbf{v}_i$ is then used to generate the code. For example, if the difference in voxel coordinates is $\Delta\mathbf{v} = (2, 1, -3)$, the resulting code according to Table 1 is *001555*.

When processing a chain of internodes, the previous internode's endpoint is used as a reference for encoding the orientation of the next internode. The internode codes are separated using termination symbols or, in due cases, interspersed with characters denoting forking points or branch tips. The forking points are registered in a queue and retrieved in the stored order upon encountering the branch tips. At every ramification, the main branch extension is followed immediately, while the lateral one is traversed when the forking point is dequeued. Fig. 4 shows an example of such encoding of a simple planar tree skeleton.

The proposed encoding supports multi-way branching by treating it as a sequence of bifurcations with intermediate zero-length internodes. When deforking to the first lateral extension that is part of a multi-way branching point, an immediate fork is, thus, appended to the chain code sequence for each of the extension's siblings (producing a symbol sequence *76...6*).

There is an inherent loss of spatial accuracy in quantization of tree geometry, because the decoded endpoint locations can only be placed within the extent of the corresponding voxel. The average reconstruction error is minimized by using the

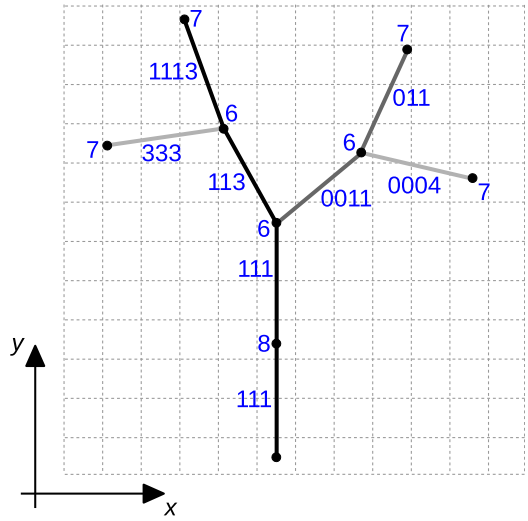

Fig. 4. Ordered encoding of internode chains: starting at the root and following the darker extension at each fork first before returning to the earliest enqueued lateral extension produces the tree descriptor *111811 16113611137001160117333700047*.

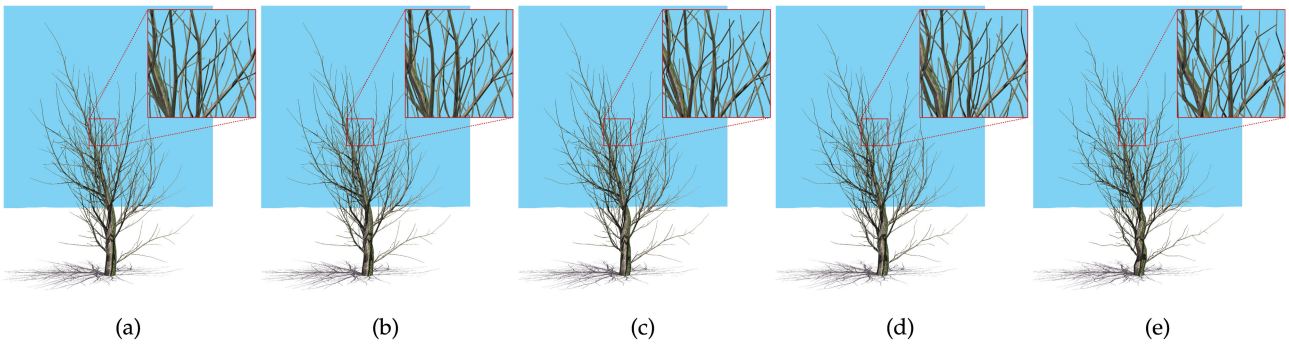

Fig. 5. Chain coding of a tree at various resolutions: (a) The original tree model has been encoded and reconstructed from chain code representations at resolution r equal to (b) 10, (c) 7, (d) 5, and (e) 3. The respective maximum and average normalized Hausdorff distances from the original model are $(\Delta_{max}/\Delta_{avg})$: (b) $3 \cdot 10^{-3}/3.7 \cdot 10^{-4}$, (c) $4.5 \cdot 10^{-3}/6.2 \cdot 10^{-4}$, (d) $9.4 \cdot 10^{-3}/1.2 \cdot 10^{-3}$, and (e) $1.1 \cdot 10^{-2}/1.7 \cdot 10^{-3}$.

voxel center as the reconstructed endpoint position, in which case the error is bounded from above by $\sqrt{3}d/2$. This upper bound can be reduced by increasing the value of resolution parameter r, which allows the balancing of accuracy and code length. For approximated geometric tree models that were obtained from remote sensing data or built using modeling (e.g., growth simulation) software, such adjustable accuracy loss is acceptable, because its effect on the model's visual properties is negligible. This can be observed in Fig. 5, which compares the original tree model to its reconstructions at different resolutions. The quality of reconstruction is reported as its maximum and average normalized (with respect to the bounding box diagonal) Hausdorff distance from the original [61].

A major improvement in code length can be achieved by taking into account the branching properties of trees, which are reflected in their geometric models. When extending during growth, tree branches usually make small turns in order to adapt to changing lighting conditions and avoid being overshadowed by the rest of the tree structure, while forces like gravity and constant wind modulate branch growth directions in a continuous way. Significant direction changes can occur at forks, particularly for lateral branches with smaller radii, while the main branch extensions usually continue in approximately the same direction. Special cases, in which a branch ostensibly makes a large turn, are those of natural branch shedding and tree pruning, where one of the extensions has been removed selectively by the plant itself or a human, respectively.

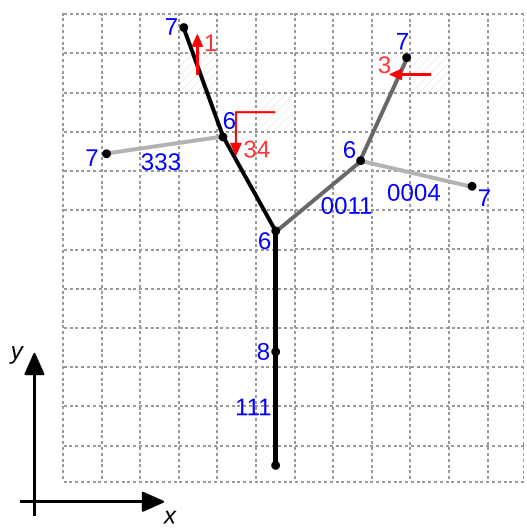

Fig. 6. Differential F6 encoding of internode chains: The last internode orientation is used to predict the terminal voxel (hatched boxes) for the next internode and only the necessary corrections (red arrows) are encoded. At the forking points, the orientation for the first internode of the lateral extension is fully encoded. This produces the tree descriptor *1118634617001163733370004*7.

The property of slowly changing internode directions along the main branches of a tree skeleton can be exploited for shortening its descriptor using a differential chain code. To this end, the skeleton needs to be traversed by following the extension with the smallest direction deviation (typically the one with the largest radius) first at all forking points, storing the references to lateral branches in a queue. In order to achieve maximum code reduction, the same principle is applied recursively at all levels. The differential chain code is then generated by encoding fully only the orientation of the first internode of every branch, while subsequent internodes are assumed to be equally oriented. If this is not the case, only the orientation correction is encoded, which, in most cases, requires much shorter code (Fig. 6). We shall call the proposed encoding a Differential Branching F6 (DBF6) chain code.

With DBF6, long straight branches can be encoded using a single symbol (i.e., the internode termination character *8*) for all except the first internode. This is an ideal case though, because, even for a sequence of collinear internodes, a correction has to be inserted occasionally in order to compensate for the accumulating quantization error. Nevertheless, substantial code length reduction can be achieved. For instance, the chain encoding of a tree from Fig. 5 at resolution $r = 5$ using the regular and differential code resulted in respective descriptor lengths of 18910 versus 10534, or 44 percent difference. The encoding procedure, which generates both the descriptor C and the corresponding ordered list of internode radii R, is shown in Pseudocode 1.

3.2 Level-of-Detail

Visualization of large afforested areas requires rendering of distant trees at lower geometric detail. It is desirable that a properly reduced tree skeleton can be derived from the full resolution model directly in its encoded form, which minimizes the necessary space for storing the complete LOD hierarchy and enables targeted LOD extraction during visualization. In the continuation, we describe a procedure for reducing the geometry of chain coded tree skeletons by culling thin branches.

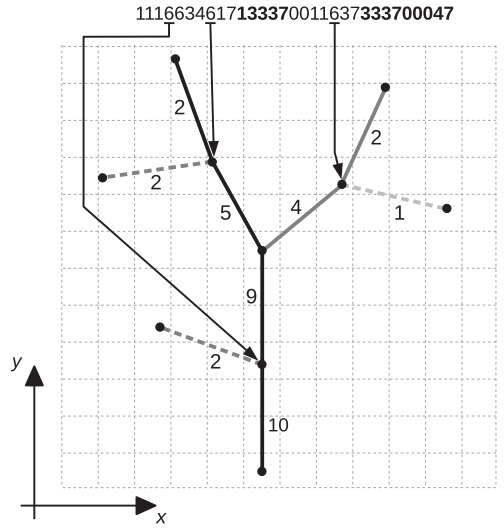

Fig. 7. LOD reduction of tree skeleton using the chain code representation: The radii values are written next to the internodes, which are shaded by order of traversal from darkest to lightest. By setting the branch thickness threshold to $T = 3$, the dashed parts of the skeleton are pruned and the bold parts of the descriptor are removed. The positions of forking points that are replaced with the internode termination symbols are tagged in the chain code and connected to their positions in the skeleton.

Pseudocode 1. DBF6 Encoding Procedure for a Given Tree Skeleton $Tree$

```
1: procedure EncodeTree (Tree)
2:   initialize empty chain code C and list of radii R
3:   enqueue root internode of Tree into queue Q
4:   while the queue Q is not empty do
5:     dequeue the next internode I from Q
6:     append I's radius to R
7:     calculate I's orientation by subtracting its endpoints
8:     if I is the first internode of a lateral branch then
9:       if this is the first extension in a multi-way fork then
10:        for each of the sibling lateral extensions do
11:          append fork symbol '6' to C
12:          enqueue sibling's first internode at the back of Q
13:      convert I's orientation to chain code and append it to C
14:    else
15:      calculate orientation correction from previous
         orientation
16:      convert the correction to chain code and append it to C
17:    if I has no successors then
18:      append defork symbol '7' to C
19:    else if I has a single successor then
20:      append internode termination symbol '8' to C
21:      enqueue the successor internode at the head of Q
22:    else if I has multiple successors then
23:      append fork symbol '6' to C
24:      enqueue the main successor at the head of Q
25:      enqueue the first lateral successor at the back of Q
26:  return (C, R)
```

When a chain code descriptor of a tree skeleton is produced using the procedure in Pseudocode 1, the internode radii are collected in a separate list, such that a positional correspondence exists between the internodes and their radii. Let a tree descriptor C, a corresponding radius sequence R, and a branch radius threshold T be given. The LOD reduction step in Pseudocode 2 produces the shortened tree descriptor C' in which all of the lateral branches, whose starting internode radii are smaller than the threshold, are pruned. The abridged list of radii R' is also generated for use in successive LOD reductions.

Pseudocode 2. LOD Reduction at Threshold Branch Radius T

```
1: procedure ReduceLOD (C, R, T)
2:   initialize empty C', R' and queue Q
3:   position at the start of C and R
4:   while not at the end of C do
5:     if Q is not empty then
6:       dequeue next fork symbol location k in C' from Q
7:     else
8:       set k ← −1
9:     if the current internode radius is below threshold T
       then
10:      % prune next branch
11:      if k ≥ 0 then          % the parent wasn't pruned
12:        replace the fork symbol in C' at index k by symbol '8'
13:      % skip the rest of the branch
14:      while the current symbol in C is not '7' do
15:        if the current symbol in C is '6' or '8' then
16:          advance R
17:        if the current symbol in C is '6' then
18:          % signal a removed parent
19:          enqueue −1 at the back of Q
20:        advance C
21:      advance C and R
22:    else          % copy the rest of the branch
23:      while the current symbol in C is not '7' do
24:        if the current symbol in C is '6' or '8' then
25:          append curr. radius from R to R' and advance R
26:        if the current symbol in C is '6' then
27:          enqueue current position in C' to the back of Q
28:        append current symbol from C to C' and advance C
29:      append current radius from R to R' and advance R
30:      append current symbol from C to C' and advance C
31:  return (C', R')
```

The main idea is to process the code string and the list of radii in parallel, checking the initial radius of each encountered lateral extension against the threshold. In case the branch should be retained, the corresponding code section and subsequence of radii are copied to the output; otherwise they are omitted (Pseudocode 2). For each of the subbranches, either the mark of a pruned parent or the new position of a corresponding fork symbol needs to be stored in a queue. In this way, all of the fork symbols corresponding to pruned branches with unpruned parents can be replaced in the code by internode termination symbols. The proposed procedure does not foreshorten the main extensions, because the principal chain of internodes is considered as a unit branch. This corresponds to the Weibull ordering scheme for trees, where the depth index is incremented only for the dominated extensions at the forking points [37].

As an example of LOD reduction, let us consider the descriptor $C = 1116634\ 617133370011637333700047$, which describes the topological structure in Fig. 7, with the

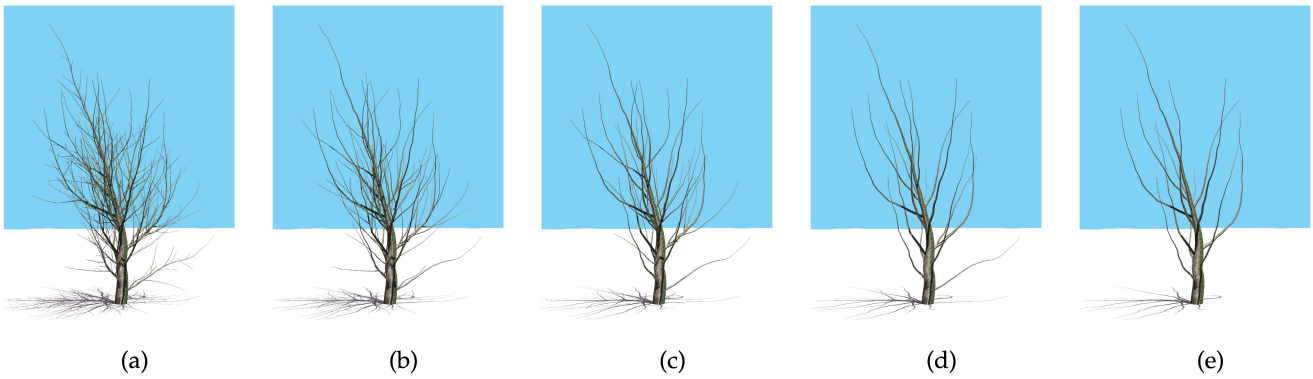

Fig. 8. Progressive LOD reduction of a tree skeleton: (a) full-detail model has been simplified by pruning the branches at radius threshold value T equal to (b) 13, (c) 24, (d) 53, and (e) 88. The respective maximum and average normalized Hausdorff distances from the full-detail model are $(\Delta_{max}/\Delta_{avg})$: (b) $9.9 \cdot 10^{-2}/5 \cdot 10^{-3}$, (c) $9.9 \cdot 10^{-2}/5.1 \cdot 10^{-3}$, (d) $1.1 \cdot 10^{-1}/1.5 \cdot 10^{-2}$, and (e) $1.2 \cdot 10^{-1}/1.9 \cdot 10^{-2}$.

corresponding sequence of radii $R = [10, 9, 5, 2, 2, 4, 2, 2, 1]$. Setting the threshold to $T = 3$ prunes the dashed parts of the skeleton and extracts the emphasized parts of C in Fig. 7. The corresponding subsequence of radii is removed from R as well, resulting in the new descriptor $C' = 11186348170011837$ and sequence of radii $R' = [10, 9, 5, 2, 4, 2]$.

Fig. 8 shows the results of progressive LOD reduction, performed on a chain code tree model representation by using an increasing branch thickness threshold.

3.3 Interpolative Compression

It has been shown that a non-decreasing sequence of integers can be compressed very efficiently using lossless interpolative encoding if it contains clusters of nearly sequential values, such as in the case of inverted indexes [58]. Interpolative encoding has also been applied to four different types of 2D chain codes in order to achieve state-of-the-art compression results [59]. In this section, we build on these findings and propose a method for efficient compression of geometric tree models that applies interpolative encoding to both the differential 3D chain code descriptor and its corresponding sequence of internode radii.

The interpolative compression of tree descriptors mainly follows the approach for 2D chain codes presented in [59], in which a given string of chain code symbols C is converted into a non-decreasing list of integers through the following steps:

1) The Burrows-Wheeler transform (BWT) [62] of the descriptor C is performed, which permutes the input string in order to obtain longer sequences of the same symbols. The outputs of BWT are the permuted string C' and a key n, which is necessary for the reconstruction.
2) The Move-To-Front (MTF) transform [63] is applied, in which the chain code symbols are replaced in C' with their positions (i.e., indexes) in the code table. While processing an input string, the MTF heuristic shifts the entries in the code table by moving the last encountered symbol to the front position. When applied to the output string of the BWT, such re-labeling produces a sequence C'' with long runs of small values, which is particularly amenable to efficient compression.
3) A cumulative sum of C'' produces the required non-decreasing sequence of integers, which is then passed to the interpolative encoder.

The motivation for applying interpolative compression to the radii sequences, produced by the tree encoding procedure in Pseudocode 1, is based on the observation that their characteristics are similar to those of an inverted index. Because the radius decreases slowly along the branch, making potentially larger jumps only at the forking points, the result is a partially ordered sequence with small internal d-gaps. In order to convert a radii list R into a non-decreasing sequence, the following succession of steps is used:

1) The radii sublists that correspond to individual branches are inverted, and a list of d-gaps is produced for each sublist.
2) The sublists are then merged by ordinary concatenation. Because there is a one-to-one mapping with the list of internodes, the boundaries between sublists can always be restored.
3) If necessary, the concatenated list is zero-aligned by subtracting the smallest radius value (usually 1). Because in a typical tree model the number of branches increases exponentially with decreasing thickness (e.g., the number of twigs with minimal radius is the highest), the resulting list contains many zero values. This requires storing of the offset value with compression metadata, but such minimal overhead is usually compensated for by the increased efficiency of the interpolative compression.
4) The cumulative sum Q of the concatenated list is computed, which generates the non-decreasing input sequence for the interpolative encoder.

Fig. 9 shows the graphs of the raw radii sequence R and the corresponding sequence Q for the tree from Fig. 5.

4 Results

In this section, we report the storage requirements when encoding tree models of varying complexity using the proposed method. The results are compared with the raw tree geometry representation, the binary encoded chain code representation, and the chain code representation compressed using the standard tools. We demonstrate that the proposed encoding and compression scheme for geometric

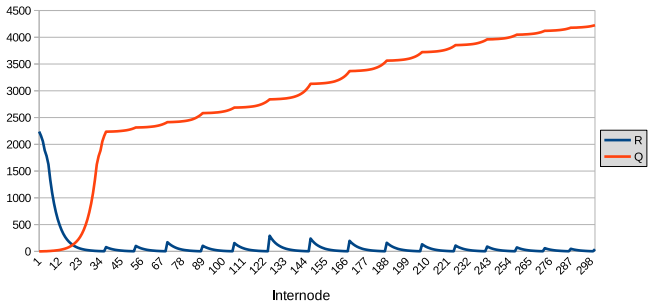

Fig. 9. The graphs of raw radius sequence R and corresponding cumulative sum sequence Q for the tree from Fig. 5. The graphs are truncated to first 300 internodes, so the clustering patterns of small d-gap values are clearly visible.

tree models produces a compact binary stream with input-to-output storage reduction factors between 8 and 14. This allows us to store the geometric description of a forest scene with 412 trees, at full detail, in around 1 MB of memory space.

4.1 Experimental Setup

In all of our experiments, the tree models generated by the GPU-based tree synthesis method presented in [36] were used. The geometric description of a tree model is stored in an array whose elements represent internodes and contain both geometrical and topological information. Radius values of individual internodes are determined as the number of buds in the chain of successors, while the actual cone radii are computed at rendering time. The tree traversal procedure is adapted to convert this representation into a pair of sequences, the chain code C and the radius list R, as described in Section 3. A C++ implementation of the method was used for determining the decompression times on a desktop computer with an Intel i7-7700K CPU, 16 GB of RAM, and a GNU/Linux OS kernel version 4.19.

TABLE 2
Properties of Tree Models from Fig. 10

Tree model	Internodes	Forks	Branch levels	Age
Fig. 10a	3394	1209	4	100
Fig. 10b	3733	1357	4	28
Fig. 10c	1114	449	6	9
Fig. 10d	8701	3243	5	100
Fig. 10e	3275	1086	4	10
Fig. 10f	865	426	7	18
Fig. 10g	634	196	5	5
Fig. 10h	553	225	5	10
Fig. 10i	859	418	6	18
Fig. 10j	3746	331	3	17
Fig. 10k	971	321	3	10
Fig. 10l	1408	574	6	15

Fig. 10 shows a selection of the tree models that were used in our experiments. They belong to diverse simulated species and were drawn from different developmental stages during growth simulation in order to obtain models of varying size and complexity. The main characteristics of these models are collected in Table 2.

In the continuation, the results of performing the following experiments are reported:

- Experiment 1: Chain coding of tree models at different resolutions and LOD levels,
- Experiment 2: Interpolative compression of full-detail tree models, and
- Experiment 3: Chain coding and compression of complete forest scene at different chain code resolutions and multiple LODs.

Experiment 1

For this test, resolution values $r = 3$, 5, and 10 were used for the chain coding of tree models from Fig. 10. The LOD

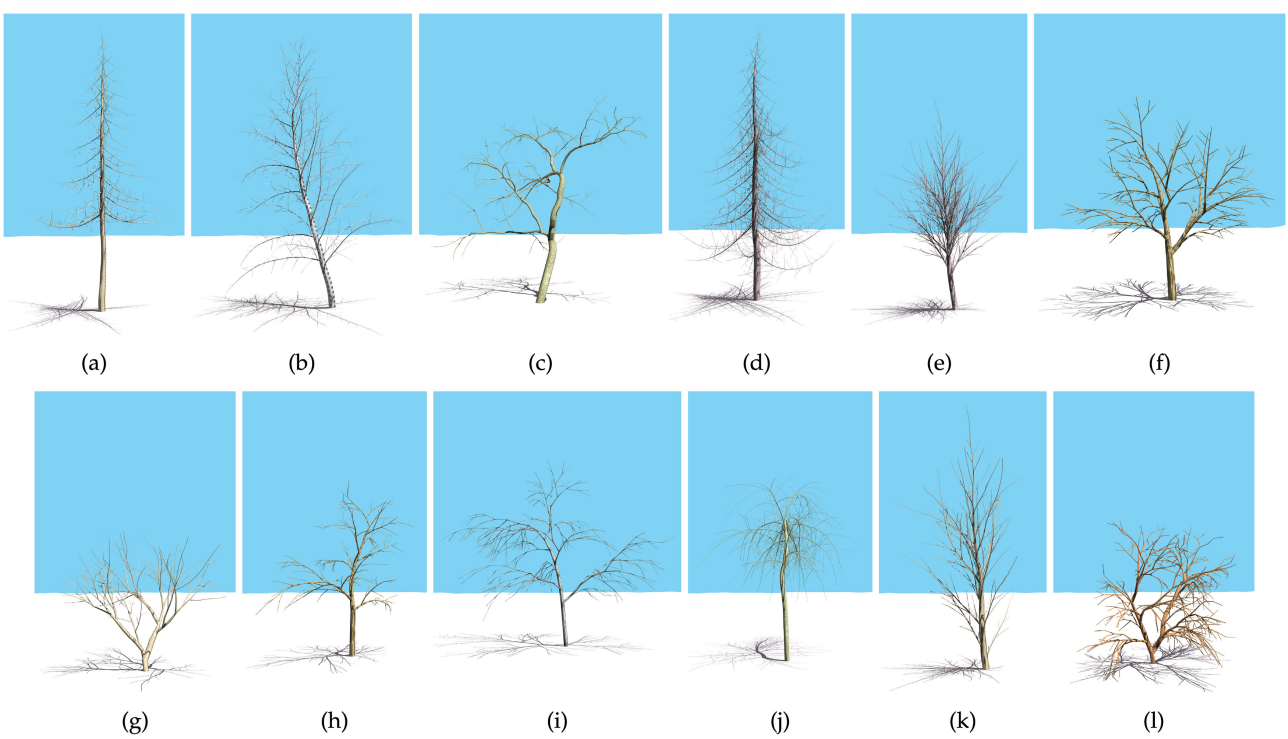

Fig. 10. Tree models used in the Experiments 1 and 2.

TABLE 3
Lengths of DBF6 Codes Obtained by Encoding the Tree Models from Fig. 10 at Three Different Resolutions and Four LODs

Tree model	Resolution		
	3	5	10
Fig. 10a	12223/1278/237/137	15874/1330/261/164	25257/1574/305/193
Fig. 10b	13401/1654/438/145	17525/1766/462/152	28529/2256/579/192
Fig. 10c	4353/817/450/161	6073/1040/572/205	10670/1622/840/295
Fig. 10d	31616/2555/1108/140	41656/2685/1167/156	66986/3076/1268/173
Fig. 10e	11624/1244/698/68	14964/1267/705/75	24131/1630/921/115
Fig. 10f	3490/533/267/37	4890/654/343/45	8437/971/484/52
Fig. 10g	2234/549/302/60	3003/700/372/60	4957/1045/551/91
Fig. 10h	2133/360/177/36	2958/455/195/44	5096/712/326/60
Fig. 10i	3542/565/260/39	4964/725/328/44	8770/1147/511/70
Fig. 10j	10472/1612/930/200	11598/1657/948/214	15428/2095/1160/245
Fig. 10k	3443/660/265/62	4396/671/278/66	7074/908/417/104
Fig. 10l	5658/648/327/27	7966/836/417/38	13885/1284/622/53

Each table cell contains code lengths in format $LOD_0/\ldots/LOD_3$.

reduction was performed using threshold values at three uniformly distributed positions in a sorted list of unique branch thicknesses, thus generating four LODs, including the full resolution level. Table 3 reports the resulting chain code lengths for different combinations of resolutions and LODs, where LOD_0 denotes the encoding of a model at full detail. Table 4 shows, in a complementary way, the relative code length reduction with respect to the ordinary F6 chain code.

The results demonstrate that the proposed encoding provides a flexible balancing mechanism between descriptor length and reconstruction accuracy through the adjustable resolution parameter. It can also be confirmed that the DBF6 chain code already allows substantial reduction of descriptor lengths with respect to the regular (i.e., non-differential) F6 code. It is not surprising that the average number of chain code symbols per internode is lower for tree models with prolonged and slowly bending extensions. Further significant reduction of descriptor lengths can be achieved by lowering the LOD. The relative savings with respect to the non-differential chain code increase on coarser LODs, because the number of internodes that have to be fully encoded decreases when thin lateral branches are removed.

Experiment 2

For this test, full-detail tree models from Fig. 10 were chain-coded at resolution $r = 5$ and then compressed using the interpolative encoding scheme from Section 3.3. Table 5 reports the total sizes of the following model representations:

- raw binary storage of tree geometry, which requires 16 bytes per internode, including the pointers.
- compressed raw binary storage of tree geometry, for which the *gzip* tool [64] was used with maximum compression settings (option *-9*);
- binary encoding of chain code representation, for which the minimal number of bits was used to store the internode code/radius pair (i.e., 4 bits per symbol for the code and as few as possible for the radius);
- *gzip*-compressed chain code representation, obtained by streaming the code and radii sequences through the *gzip* compressor at maximum compression setting;

TABLE 4
Relative Reduction (in %) of DBF6 Code Length with Respect to F6 When Encoding the Tree Models from Fig. 10 at Three Different Resolutions and Four LODs Using the Proposed Differential Code

Tree model	Resolution		
	3	5	10
Fig. 10a	35/52/59/65	45/68/71/73	54/80/82/83
Fig. 10b	35/52/52/52	45/67/67/67	52/78/78/78
Fig. 10c	29/46/45/46	36/56/55/55	40/63/65/66
Fig. 10d	35/53/54/61	45/68/69/71	53/81/82/83
Fig. 10e	36/48/50/59	47/66/67/70	54/77/77/76
Fig. 10f	27/43/47/51	34/55/56/61	39/65/67/76
Fig. 10g	36/46/46/45	45/55/57/65	52/65/66/72
Fig. 10h	30/43/43/50	37/54/60/60	42/62/64/71
Fig. 10i	27/43/44/50	33/53/55/64	38/60/63/70
Fig. 10j	49/52/52/53	64/68/68/67	75/79/80/80
Fig. 10k	35/48/52/58	47/66/67/71	54/75/74/76
Fig. 10l	28/42/41/45	34/52/52/52	39/61/62/64

Each table cell contains code lengths in format $LOD_0/\ldots/LOD_3$.

TABLE 5
Sizes (in Bytes) of Full-Detail Tree Models from Fig. 10 Using Raw (*raw*) and *gzip*-Compressed Raw (*raw gzip*) Tree Geometry, Binary Encoded (*cc bin*), *Gzip*-Compressed (*cc gzip*), and Interpolatively Compressed Chain Code Representation (*cc ic*) at Resolution $r = 5$

Tree model	raw	raw gzip	cc bin	cc gzip	cc ic	Dec. time
Fig. 10a	54304	48535	13474	6256	5124 (10.5)	0.59
Fig. 10b	59728	52852	14850	6876	5701 (10.4)	0.64
Fig. 10c	17824	15563	4589	2548	2070 (8.6)	0.22
Fig. 10d	139216	125224	36076	15395	12884 (10.8)	1.50
Fig. 10e	52400	46742	12416	4939	4715 (11.1)	0.53
Fig. 10f	13840	12047	3548	1843	1551 (8.9)	0.17
Fig. 10g	10144	8927	2315	1384	1089 (9.3)	0.11
Fig. 10h	8848	7789	2192	1288	1018 (8.6)	0.11
Fig. 10i	13744	11943	3577	1842	1550 (8.8)	0.17
Fig. 10j	59936	53260	11908	4573	4280 (14.0)	0.49
Fig. 10k	15536	13782	3555	1684	1458 (10.6)	0.16
Fig. 10l	22528	19850	5941	3266	2607 (8.6)	0.29

The values in parentheses are compression factors of cc ic *with respect to* raw. *The last column shows decompression times in milliseconds.*

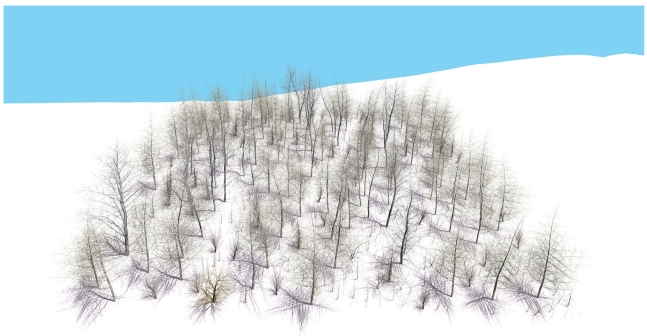

Fig. 11. The forest scene used in Experiment 3.

TABLE 6
Storage Requirements and Decompression Times for the Forest Scene from Fig. 11 with Different Combinations of Chain Code Resolution and Number of Stored LOD Levels

Number of LODs	Resolution		
	3	5	10
1	0.96 MB (105 ms)	1.07 MB (129 ms)	1.28 MB (179 ms)
3	1.17 MB (123 ms)	1.29 MB (150 ms)	1.52 MB (205 ms)
5	1.40 MB (145 ms)	1.54 MB (175 ms)	1.80 MB (235 ms)

The first row corresponds to storing the full-detail LOD only, while the second (third) row corresponds to storing the full-detail LOD plus two (four) reduced LODs.

- interpolative encoding of the chain code and radii sequences.

The results show that, by using the interpolative compression of both the resulting tree descriptors and the corresponding radii sequences, compression ratios of up to 14 with respect to the raw geometry storage can be achieved. The proposed approach is also considerably more efficient than the raw binary encoding or the *gzip*-based compression of the chain code representation. Included in Table 5 are decompression times for individual tree models, which demonstrate that on-demand decompression and decoding of tree geometry is feasible in real-time.

Experiment 3

Experiment 3 is based on a practical scenario, in which a simulated forest patch containing tree models of varying shapes and sizes is encoded and compressed using the proposed method. Fig. 11 shows an aerial view of the test forest scene used in the final experiment. The scene was produced by running the forest growth simulation within a constrained area for 30 years. It contains 412 trees with an average count of 1792 internodes per tree and total internode count 738581, where the small plants with less than 50 internodes were removed.

The LOD of trees was encoded at different chain code resolutions using 1 (i.e., full-detail model only), 3, and 5 degrees of LOD. The experiment results are summarized in Table 6, where the storage space and decompression time required for interpolatively encoded forest geometry are reported. It should be noted that, in order to allow full scene reconstruction, some extra storage is required for keeping tree locations and orientations. However, if the trees' local coordinate systems are axis-aligned with the global coordinate system and the terrain configuration is known, which are both reasonable assumptions, then only the tree's longitude and latitude need to be recorded. For the scene in Fig. 11, this corresponds to 3.2 KB, or approximately 0.3 percent of additional space.

The results of the experiment demonstrate that, even if the tree models are stored at multiple preprocessed levels of detail, the geometry of the test forest can be stored in well under 2 MB. Alternatively, only full-detail models could be encoded and the LOD derivation performed on demand. In that case, a similar forest consisting of one million trees would, depending on the encoding resolution, require between 2.2 and 3 GB of storage, which would still fit into the memory of a contemporary desktop computer. Additionally, the decompression and decoding of the forest to maximum detail takes approximately 130 ms at resolution $r = 5$, which renders the method suitable for use in interactive visualization.

5 Discussion

The results of the experiments have shown that the proposed method allows efficient storage of complex and versatile tree ramifications. The main limitation of the method is its dependence on skeletal geometric representation of tree models as chains of internodes. When the original model representation does not provide such a skeleton, it needs to be obtained by model preprocessing, which may incur accuracy loss with respect to the original.

While the method can be used to encode tree models with variable internode lengths, the encoding is most efficient when internodes are of approximately the same length. For tree models with widely varying internode lengths, the errors in the predicted endpoint positions of the following internodes would potentially increase and produce longer correction codes. In such cases, a preliminary subdivision of skeleton geometry may improve the efficiency of the differential chain coding. Chain coding is also less efficient for shrub-like models with frequent forks and short lateral branches, because the first internode of each lateral branch is always fully encoded. On the other hand, tree models with prominent apical dominance achieve better compression, because the branch directions and radii change less at the forks.

The presented method allows compression of tree skeletons and does not address the storage of tree foliage. A separate application of foliage simplification methods is, therefore, required for LOD processing and storage of foliage geometry. While the method could be extended to encode leaves (e.g., as a special type of internodes), storing the exact positions and orientations of leaves is not needed for many applications and would be prohibitively expensive in situations with thousands or millions of trees. In such cases, tree foliage can be generated on-the-fly at locations determined by the reconstructed branching structure, and with reproducibly random orientations (e.g., as in [40]).

The chain code-based representation of tree models is similar to other structure encoding schemes, such as L-systems [34] or iterated function systems (IFS) [65]. Each approach uses a tree descriptor, which is decoded into an actual geometric model by an interpreting procedure. With L-systems, the descriptor is a list of production rules, with IFS it is a set

of contractions, and with chain code it is a sequence of code symbols. The main difference between these approaches is that the chain codes are used to encode preexisting tree models of general shapes, while the L-systems and the IFS can be used more directly as methods for synthesis of tree models that are constrained by a given set of rules.

The reported decompression times indicate that the method is suitable for incorporation in interactive visualization applications to perform on-demand decoding of tree models. An integration of the procedure with the system for management and caching of geometry in different stages of decompression is a prospective direction for further research.

6 Conclusion

In the future, increasingly detailed modeling of the environment can be expected, which includes vegetation and trees in particular. In order to detect, simulate and predict vegetation changes reliably, the data should be collected, visualized and analyzed repeatedly, which requires efficient representation and storage of geometric models. In this paper, these issues are addressed through the following steps:

- a differential chain code-based representation of geometric tree models is introduced, which encodes orientation changes of successive tree skeleton elements;
- a procedure for derivation of tree models with reduced level of detail directly from the chain code representation is defined;
- an efficient mechanism for combined interpolative compression of tree descriptors and internode radii sequences is proposed.

A potentially promising direction for future research is to incorporate other base chain codes with the process of orientation change encoding. Further optimization in the compression of radii sequences is also possible, e.g., by considering their non-linear properties that originate in the natural laws of tree physiology.

Acknowledgments

This work was supported by Slovenian Research Agency (research program P2-0041 and project J2-8176).

References

[1] Z. Zhu and C. E. Woodcock, "Continuous change detection and classification of land cover using all available Landsat data," *Remote Sensing Environment*, vol. 144, pp. 152–171, 2014.
[2] X. Zhang, P. Xiao, X. Feng, and M. Yuan, "Separate segmentation of multi-temporal high-resolution remote sensing images for object-based change detection in urban area," *Remote Sens. Environment*, vol. 201, pp. 243–255, 2017.
[3] V. M. Gómez-Muñoz, M. A. Porta-G ándara, and J. L. Fernández, "Effect of tree shades in urban planning in hot-arid climatic regions," *Landscape Urban Planning*, vol. 94, no. 3–4, pp. 149–157, 2010.
[4] K. J. Anderson-Teixeira, J. C. McGarvey, H. C. Muller-Landau, J. Y. Park, E. B. Gonzalez-Akre, V. Herrmann, A. C. Bennett, C. V. So, N. A. Bourg, J. R. Thompson, et al., "Size-related scaling of tree form and function in a mixed-age forest," *Functional Ecology*, vol. 29, no. 12, pp. 1587–1602, 2015.
[5] R. W. Miller, R. J. Hauer, and L. P. Werner, *Urban Forestry: Planning and Managing Urban Greenspaces*, 3rd ed. Long Grove, IL, USA: Waveland Press, 2015.
[6] Z. Tan, K. K.-L. Lau, and E. Ng, "Urban tree design approaches for mitigating daytime urban heat island effects in a high-density urban environment," *Energy Buildings*, vol. 114, pp. 265–274, 2016.
[7] T. Van de Peer, K. Verheyen, V. Kint, E. Van Cleemput, and B. Muys, "Plasticity of tree architecture through interspecific and intraspecific competition in a young experimental plantation," *Forest Ecology Manag.*, vol. 385, pp. 1–9, 2017.
[8] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan, "Image-based tree modeling," *ACM Trans. Graph.*, vol. 26, pp. 87:1–87:7, 2007.
[9] H. Xu, N. Gossett, and B. Chen, "Knowledge and heuristic-based modeling of laser-scanned trees," *ACM Trans. Graph.*, vol. 26, no. 4, pp. 19:1–19:13, 2007.
[10] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-Sana, "Automatic reconstruction of tree skeletal structures from point clouds," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 151:1–151:8, 2010.
[11] M. Rutzinger, A. K. Pratihast, S. J. Oude Elberink, and G. Vosselman, "Tree modelling from mobile laser scanning data-sets," *Photogrammetric Record*, vol. 26, no. 135, pp. 361–372, 2011.
[12] P. Raumonen, M. Kaasalainen, M. Åkerblom, S. Kaasalainen, H. Kaartinen, M. Vastaranta, M. Holopainen, M. Disney, and P. Lewis, "Fast automatic precision tree models from terrestrial laser scanner data," *Remote Sens.*, vol. 5, no. 2, pp. 491–520, 2013.
[13] V. Méndez, J. R. Rosell-Polo, R. Sanz, A. Escolà, and H. Catalán, "Deciduous tree reconstruction algorithm based on cylinder fitting from mobile terrestrial laser scanned point clouds," *Biosystems Eng.*, vol. 124, pp. 78–88, 2014.
[14] Z. Wang, L. Zhang, T. Fang, P. T. Mathiopoulos, H. Qu, D. Chen, and Y. Wang, "A structure-aware global optimization method for reconstructing 3-D tree models from terrestrial laser scanning data," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 9, pp. 5653–5669, Sep. 2014.
[15] A. Kato, H. Obanawa, Y. Hayakawa, M. Watanabe, Y. Yamaguchi, and T. Enoki, "Fusion between UAV-SFM and terrestrial laser scanner for field validation of satellite remote sensing," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2015, pp. 2642–2645.
[16] J. Guo, S. Xu, D.-M. Yan, Z. Cheng, M. Jaeger, and X. Zhang, "Realistic procedural plant modeling from multiple view images," *IEEE Trans. Vis. Comput. Graph.*, 2019, doi: 10.1109/TVCG.2018.2869784.
[17] W. Pałubicki, K. Horel, S. Longay, A. Runions, B. Lane, R. Měch, and P. Prusinkiewicz, "Self-organizing tree models for image synthesis," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 58:1–58:10, 2009.
[18] S. Pirk, O. Šťava, J. Kratt, M. A. Massih Said, B. Neubert, R. Měch, B. Beneš, and O. Deussen, "Plastic trees: Interactive self-adapting botanical tree models," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 50:1–50:10, 2012.
[19] G. Wang, H. Laga, J. Jia, N. Xie, and H. Tabia, "Statistical modeling of the 3D geometry and topology of botanical trees," *Comput. Graph. Forum*, vol. 37, no. 5, pp. 185–198, 2018.
[20] J. Lluch, E. Camahort, and R. Vivó, "Procedural multiresolution for plant and tree rendering," in *Proc. 2nd Int. Conf. Comput. Graph. Virtual Reality Vis. Interaction Africa*, 2003, pp. 31–38.
[21] A. Jonas and N. Kiryati, "Digital representation schemes for 3D curves," *Pattern Recognit.*, vol. 30, no. 11, pp. 1803–1816, 1997.
[22] E. Bribiesca, "A chain code for representing 3D curves," *Pattern Recognit.*, vol. 33, no. 5, pp. 755–765, 2000.
[23] T. Wang, I. Cheng, V. Lopez, E. Bribiesca, and A. Basu, "Valence normalized spatial median for skeletonization and matching," in *Proc. IEEE 12th Int. Conf. Comput. Vis. Workshops*, 2009, pp. 55–62.
[24] E. Bribiesca, A. Guzmán, and L. A. Martínez, "Enclosing trees," *Pattern Anal. Appl.*, vol. 15, no. 1, pp. 1–17, 2012.
[25] L. A. Martínez, E. Bribiesca, and A. Guzmán, "Chain coding representation of voxel-based objects with enclosing, edging and intersecting trees," *Pattern Anal. Appl.*, vol. 20, no. 3, pp. 825–844, 2017.
[26] E. Bribiesca, "A method for representing 3D tree objects using chain coding," *J. Visual Commun. Image Representation*, vol. 19, no. 3, pp. 184–198, 2008.
[27] W. Aguilar and E. Bribiesca, "Symmetry detection in 3D chain coded discrete curves and trees," *Pattern Recognit.*, vol. 48, no. 4, pp. 1420–1439, 2015.
[28] H. Sánchez-Cruz, H. H. López-Valdez, and F. J. Cuevas, "A new relative chain code in 3D," *Pattern Recognit.*, vol. 47, no. 2, pp. 769–788, 2014.
[29] S. Kolmanič, N. Guid, and J. Diaci, "ForestMAS – a single tree based secondary succession model employing Ellenberg indicator values," *Ecological Modelling*, vol. 279, pp. 100–113, 2014.

[30] J. Weber and J. Penn, "Creation and rendering of realistic trees," in *Proc. 22nd Annu. Conf. Comput. Graphics Interactive Techn.*, 1995, pp. 119–128.
[31] Y. Rodkaew, P. Chongstitvatana, S. Siripant, and C. Lursinsap, "Particle systems for plant modeling," *Plant Growth Modeling Appl.*, pp. 210–217, 2003.
[32] B. Beneš, N. Andrysco, and O. Šťava, "Interactive modeling of virtual ecosystems," in *Proc. 5th Eurographics Conf. Natural Phenom.*, 2009, pp. 9–16.
[33] Q. Deng, X. Zhang, G. Yang, and M. Jaeger, "Multiresolution foliage for forest rendering," *Comput. Animation Virtual Worlds*, vol. 21, no. 1, pp. 1–23, 2010.
[34] F. Boudon, C. Pradal, T. Cokelaer, P. Prusinkiewicz, and C. Godin, "L-Py: An L-system simulation framework for modeling plant architecture development based on a dynamic language," *Frontiers Plant Sci.*, vol. 3, pp. 76:1–76:20, 2012.
[35] S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz, "Treesketch: Interactive procedural modeling of trees on a tablet," in *Proc. Int. Symp. Sketch-Based Interfaces Modeling*, 2012, pp. 107–120.
[36] Š. Kohek and D. Strnad, "Interactive synthesis of self-organizing tree models on the GPU," *Comput.*, vol. 97, no. 2, pp. 145–169, 2015.
[37] M. Holton, "Strands, gravity and botanical tree imagery," *Comput. Graph. Forum*, vol. 13, no. 1, pp. 57–67, 1994.
[38] B. Neubert, T. Franken, and O. Deussen, "Approximate image-based tree-modeling using particle flows," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 88:1–88:8, 2007.
[39] L. Xu and D. Mould, "A procedural method for irregular tree models," *Comput. Graph.*, vol. 36, no. 8, pp. 1036–1047, 2012.
[40] Š. Kohek and D. Strnad, "Interactive large-scale procedural forest construction and visualization based on particle flow simulation," *Comput. Graph. Forum*, vol. 37, no. 1, pp. 389–402, 2018.
[41] N. MacDonald, *Trees and Netw. Biological Models*. New York, NY, USA: Wiley, 1983.
[42] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Trans. Electron. Comput.*, vol. 10, no. 2, pp. 260–268, 1961.
[43] E. Bribiesca, "A new chain code," *Pattern Recognit.*, vol. 32, no. 2, pp. 235–251, 1999.
[44] H. Sánchez-Cruz and R. M. Rodr íguez-Dagnino, "Compressing bilevel images by means of a three-bit chain code," *Opt. Eng.*, vol. 44, no. 9, pp. 1–8, 2005.
[45] B. Žalik, D. Mongus, Y.-K. Liu, and N. Lukač, "Unsigned Manhattan chain code," *J. Visual Commun. Image Representation*, vol. 38, pp. 186–194, 2016.
[46] E. Lemus, E. Bribiesca, and E. Garduño, "Representation of enclosing surfaces from simple voxelized objects by means of a chain code," *Pattern Recognit.*, vol. 47, no. 4, pp. 1721–1730, 2014.
[47] D. P. Luebke, *Level of Detail for 3D Graphics*. San Francisco, CA, USA: Morgan Kaufmann, 2003.
[48] X. Zhang, G. Bao, W. Meng, M. Jaeger, H. Li, O. Deussen, and B. Chen, "Tree branch level of detail models for forest navigation," *Comput. Graph. Forum*, vol. 36, no. 8, pp. 402–417, 2017.
[49] Y. Livny, S. Pirk, Z. Cheng, F. Yan, O. Deussen, D. Cohen-Or, and B. Chen, "Texture-lobes for tree modelling," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 53:1–53:10, 2011.
[50] X. Zhang and F. Blaise, "Progressive polygon foliage simplification," in *Proc. Int. Symp. Plant Growth Model. Simul. Vis. their Appl.*, 2003, pp. 182–193.
[51] J. Gumbau, M. Chover, I. Remolar, and C. Rebollo, "View-dependent pruning for real-time rendering of trees," *Comput. Graph.*, vol. 35, no. 2, pp. 364–374, 2011.
[52] J. Beaudoin and J. Keyser, "Simulation levels of detail for plant motion," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2004, pp. 297–304.
[53] T. K. Heok and D. Daman, "A review on level of detail," in *Proc. Int. Conf. Comput. Graph., Imag. Vis.*, 2004, pp. 70–75.
[54] B. Neubert, S. Pirk, O. Deussen, and C. Dachsbacher, "Improved model-and view-dependent pruning of large botanical scenes," *Comput. Graph. Forum*, vol. 30, no. 6, pp. 1708–1718, 2011.
[55] G. Bao, H. Li, X. Zhang, and W. Dong, "Large-scale forest rendering: Real-time, realistic, and progressive," *Comput. Graph.*, vol. 36, no. 3, pp. 140–151, 2012.
[56] H. Samet, "Region representation: Quadtrees from boundary codes," *Commun. ACM*, vol. 23, no. 3, pp. 163–170, 1980.
[57] Z. Chen and I.-P. Chen, "A simple recursive method for converting a chain code into a quadtree with a lookup table," *Image Vis. Comput.*, vol. 19, no. 7, pp. 413–426, 2001.
[58] A. Moffat and L. Stuiver, "Binary interpolative coding for effective index compression," *Inf. Retrieval*, vol. 3, no. 1, pp. 25–47, 2000.
[59] B. Žalik, D. Mongus, N. Lukač, and K. R. Žalik, "Efficient chain code compression with interpolative coding," *Inf. Sci.*, vol. 439, pp. 39–49, 2018.
[60] P. G. Howard and J. S. Vitter, "Fast and efficient lossless image compression," in *Proc. Data Compression Conf.*, 1993, pp. 351–360.
[61] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," *Comput. Graph. Forum*, vol. 17, no. 2, pp. 167–174, 1998.
[62] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," Digital Equipment Corporation, Palo Alto, California, Tech. Rep. 124, 1994.
[63] B. Y. Ryabko, "Data compression by means of a "book stack"," *Problemy Peredachi Informatsii*, vol. 16, no. 4, pp. 16–21, 1980.
[64] [Online]. Available: https://www.gnu.org/software/gzip/, Last Accessed: Sep. 2018
[65] D. Strnad and N. Guid, "Modeling trees with hypertextures," *Comput. Graph. Forum*, vol. 23, no. 2, pp. 173–187, 2004.

Damjan Strnad received the MSc and PhD degrees in computer science from the University of Maribor, in 2000 and 2006, respectively. He is currently works as an associate professor of Computer Science at the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia. His research interests include computer graphics, artificial intelligence, and optimization algorithms.

Štefan Kohek received BSc and MEng degrees in computer science from the University of Maribor, Slovenia, in 2010 and 2012, respectively. He currently works as a teaching assistant in the Laboratory for Geometric Modeling and Multimedia Algorithms at the Faculty of Electrical Engineering and Computer Science. His research interests include computer graphics, tree growth simulation, and parallel computing.

Andrej Nerat received the BSc degree in computer science, in 2004. He is currently working in the Laboratory for Geometric Modelling and Multimedia Algorithms at Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests include computer graphics and artificial intelligence.

Borut Žalik received the BSc degree in electrical engineering, in 1985, the MSc and PhD degrees in computer science, in 1989 and 1993, respectively. He is a professor of Computer Science at University of Maribor, Slovenia. He is the head of the Laboratory for Geometric Modelling and Multimedia Algorithms at Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests includes processing of multimedia data, data compression, and computational geometry.