Terrain and Model Queries Using Scalar Representations With Wavelet Compression

Christoph Fünfzig, Torsten Ullrich, Dieter W. Fellner, and Edward N. Bachelder

***Abstract*—In this paper, we present efficient height/distance field data structures for line-of-sight (LOS) queries on terrains and collision queries on arbitrary 3-D models. The data structure uses a pyramid of quad-shaped regions with the original height/distance field at the highest level and an overall minimum/maximum value at the lower levels. The pyramid can compactly be stored in a wavelet-like decomposition but using max and plus operations. Additionally, we show how to get minimum/maximum values for regions in a wavelet decomposition using real algebra. For LOS calculations, we compare with a kd-tree representation containing the maximum height values. Furthermore, we show that the LOS calculation is a special case of a collision detection query. Using our wavelet-like approach, even general and arbitrary collision detection queries can efficiently be answered.**

***Index Terms*—Collision detection, distance fields, heightfield interrogation, kd-tree data structure, line-of-sight (LOS) computation, pyramid algorithms, wavelets.**

I. Introduction

THE SIMULATION and planning of realistic movements governed by physics is necessary for many applications. The simulation of movements is often based on collision detection, as changes of movements occur at events, where geometries collide.

Planning solutions like signal strength prediction [1], [2] and antenna placement [3] may use ray-casting models for electromagnetic wave propagation. In these models, the main propagation paths are evaluated by rays (geometric optics), and special wave effects are added at the intersection points of the rays with the terrain surface. Of course, the line-of-sights (LOSs) from the antennas account for the main propagation paths. Thus, efficient LOS computation is an important means for terrain interrogation.

In this paper, we present efficient height/distance field data structures for LOS queries on terrains and collision queries on arbitrary 3-D models. The data structure uses a pyramid of quad-shaped regions with the original height/distance field at the highest level and maximum values of regions at the lower levels (*maximum mipmap heightfield*). The pyramid can compactly be stored in a wavelet-like decomposition but using max and plus operations. Additionally, we show how to get minimum/maximum values for regions in a nonstandard wavelet decomposition in real algebra (*wavelet heightfield*). For heightfields (like regular-grid terrains), the data structure is similar to a 2-D kd-tree with added minimum/maximum height values in inner nodes.

The technique for a single heightfield can be extended to a spherical distance field for arbitrary 3-D models. With the spherical distance field according to a fixed center point, conservative distance bounds to the model can efficiently be queried. Furthermore, we show that even general and arbitrary collision detection queries can similarly be answered to LOS queries.

Applications: Height/distance field data sets are required in various applications. Due to their size, they are commonly compressed using a wavelet basis. Fig. 1 shows an example of a distance data set, compressed with the Daubechies-4 wavelet and its subband coefficients adaptively quantized. The motivation for this work is to directly use the compressed data sets (without a complete decompression step) for queries (collision and LOS tests). We call this *on-the-fly decompression during querying*, which is similar to the coupling of decompression with rendering [4]. The benefits are much lower memory requirements at the same or even smaller runtime. The problem of how to get minimum/maximum values for representations with wavelets in real algebra is covered in Section III-B. Using a special Haar wavelet-like decomposition in max-plus algebra, the maximum values of quad-shaped subregions are directly available from the representation. Due to their small support and discontinuity, they are worse in terms of compression efficiency but are very fast to reconstruct and query. The construction and reconstruction of the maximum mipmap heightfields are described in Section III-A.

II. Related Work

Basic data structures used for the LOS computation problem are the grid structure and several tree-structured schemes. The grid structure is a sampled representation, directly storing height/distance values to a reference plane or reference surface. There have been several approaches of compressing the grid data, among them spatial hashing [5] and wavelet transform [4]. The kd-tree is a binary tree with splits cycling through the d dimensions. It has been invented for organizing point sets for

Manuscript received January 30, 2008; revised June 29, 2008. The Associate Editor coordinating the review process for this paper was Dr. Annamaria Varkonyi-Koczy.

C. Fünfzig is with the CAGD Group, Univ. Valenciennes et du Hainaut-Cambrésis, FR CNRS 2956, 59300 Valenciennes, France (e-mail: c.fuenfzig@gmx.de).

T. Ullrich is with the Institute of Computer Graphics and Knowledge Visualization, Graz University of Technology, 8010 Graz, Austria (e-mail: t.ullrich@cgv.tugraz.at).

D. W. Fellner is with the Department of Graphisch-Interaktive Systeme (GRIS), Darmstadt University of Technology (TU Darmstadt), 64283 Darmstadt, Germany (e-mail: d.fellner@igd.fhg.de).

E. N. Bachelder is with Systems Technology Inc., Hawthorne, CA 90250 USA (e-mail: edbach@systemstech.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIM.2009.2016879

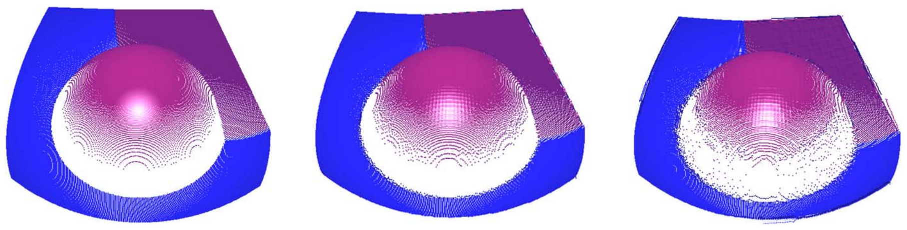


Fig. 1. Reconstruction of point cloud (256 $\times$ 256) from Daubechies-4 wavelet representations. The scene consists of a sphere in the center and a plane in the back right. (Left to right) 32-bit quantization in all subbands (MSE $= 1.92876e-019$; MAXSE $= 3.78111e-018$); 32 bits in bands 0–3, 16 bits in bands 4 and 5, 8 bits in band 6, and 4 bits in band 7 (MSE $= 8.36761e-005$; MAXSE $= 0.0261943$); and 32 bits in bands 0 and 1, 16 bits in bands 2 and 3, 8 bits in bands 4 and 5, 4 bits in band 6, and 2 bits in band 7 (MSE $= 0.00229742$; MAXSE $= 0.78619$). The MSE is the mean square error, i.e., the mean of square differences between the original and the reconstructed distance values over the full image. The MAXSE is the maximum of all square difference values.

nearest neighbor searching [6] and stores data for rectangular spatial cells in its nodes. Frisken *et al.* [7] use simultaneous splits of all dimensions resulting in a 2^d-ary tree and store data for the cell corners. With bilinear interpolation on the space region, they represent a continuous scalar function. Lefebvre and Hoppe [8] covers the problem of encoding the tree topology and the tree data with a combination of techniques suitable for random access.

LOS Calculation: The most straightforward approach to calculate LOS traverses the projection of the ray on the domain plane and checks the ray heights against the heightfield heights at a number of points per cell [9]. Different terrain reconstruction is possible for noninteger grid positions: point reconstruction, double linear interpolation, and bilinear interpolation. Without a specific test for the reconstruction used, an exact LOS test is not possible. By checking a number p of discrete points per grid cell, large low-height regions cannot be exploited. The approach always requires $l \cdot p$ height evaluations regardless of the terrain traversed, where l is the ray length in cells.

Recently, graphics processing units (GPU) have also been used to determine the LOS on a terrain (see [10] and [11]). These approaches render both the terrain surface and the ray line and test if all line fragments are above the terrain surface. If this is the case, it is a LOS up to the image resolution used for rendering and for terrain reconstruction. A special hardware occlusion test is used for counting the number of visible fragments.

There is also work on slightly extended visibility problems like computing the horizon for each grid point and direction sector [12]–[14]. Originally, the resulting horizon map was invented for self-shadowing the terrain surface.

Collision Detection: Naive approaches to determine all collisions of n object parts require a runtime of $O(n^2)$, which will rapidly be too much in practice. Innumerable algorithms with reduced runtime complexity have been developed in the last decades. They can be classified according to different schemes with regard to, e.g., field of application or solution strategy.

A general approach for collision detection uses hierarchies of simple bounding volumes containing model parts. Researchers have proposed several bounding volumes, including spheres, axis-aligned bounding boxes, oriented bounding boxes, and discrete orientation polytopes. Here, the performance depends on the tightness of the bounding volume, the efficiency of the intersection test for the bounding volume, and the strategy for hierarchy generation [15], [16].

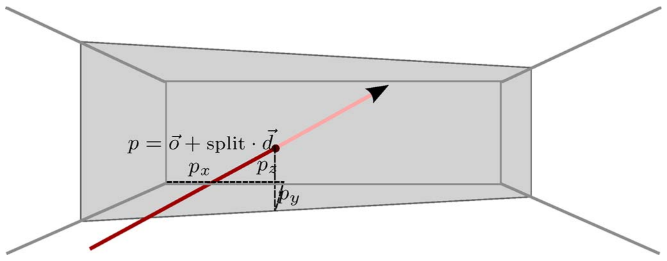


Fig. 2. Partitioning plane subdividing the ray into a near segment and a far segment at the ray–plane intersection point.

Bounding volume hierarchies with deformable models require additional time for refitting. Several papers discuss optimizations for special deformations like linear morphing, linear blend skinning, and linear combined displacement fields [17]. In this area, simpler ones like Cartesian grids and 1-D arrays accessed by hashing are commonly used. McNeely *et al.* [18] and Teschner *et al.* [5] build a voxel grid for the static model, where the points of a small movable model are queried against. This approach guarantees the high feedback rate needed by a haptic feedback device, and it is tailored to the haptic application domain.

Several authors [19], [20] use Cartesian distance fields for the implicit representation of models.

III. LOS Calculation Based on KD-Trees

A ray is defined by an origin point and a direction vector, which implicitly gives a search order on the data domain traversed by the ray. Thus, if the data domain is partitioned into several parts, then these parts can be searched according to the ray. *LOS computation* is a problem of this kind. Additionally, the parameter interval of the ray inside a domain part is available during traversal.

The simplest partition tree for a 2-D domain is a binary tree with domain-orthogonal partitioning planes. The parameter interval $[0, \infty]$ is subdivided by the ray–plane intersection point at parameter $\text{split} = \vec{n} \cdot (\vec{p} - \vec{o}) / \vec{n} \cdot \vec{d}$ into the near interval $[0, \text{split}]$ and the far interval $[\text{split}, \infty]$ (Fig. 2).

For axis-orthogonal partitioning planes, the ray–plane intersection computation $\text{split} = (p_x - o_x)/d_x$ resp. $\text{split} = (p_y - o_y)/d_y$ is simple and, therefore, particularly fast to compute. The resulting partitioning tree with alternating x- and y-orthogonal planes is called a *kd-tree* of dimension 2. The

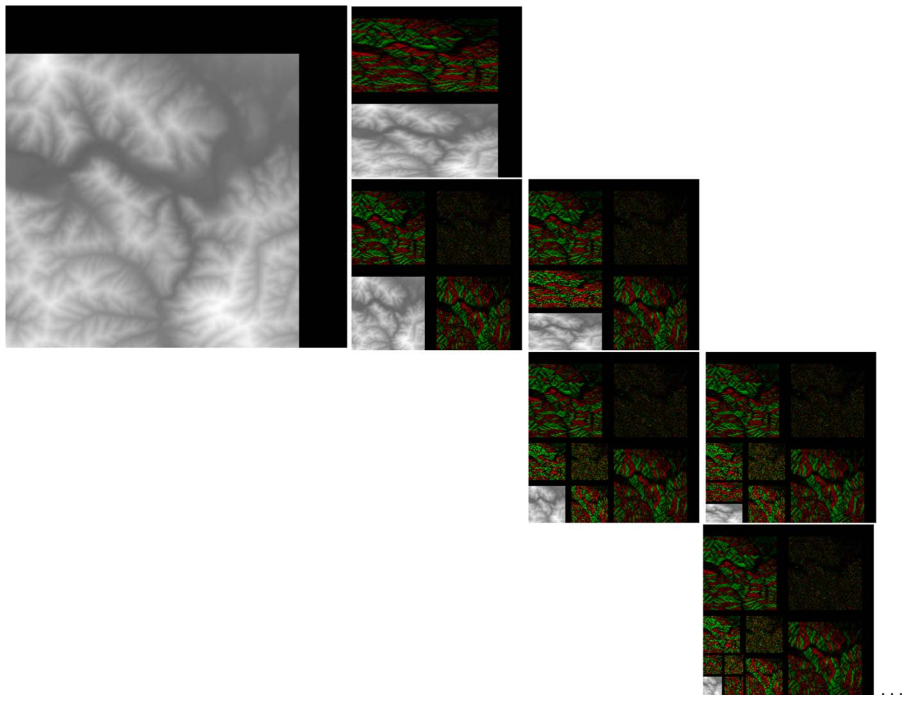


Fig. 3. (Left) Two-dimensional heightfield shown as a grayscale image. Its nonstandard decomposition using a maximum operation. Green pixels mark positive detail coefficients d_k^j, and red pixels mark negative coefficients d_k^j. A pixel's intensity encodes the difference magnitude (consequently, small differences are in black).

traversal of the kd-tree can be adapted for ray segments restricted to a parameter interval $[t_{\text{near}}, t_{\text{far}}]$. The traversal visits the near node iff $[0, \text{split}] \cup [t_{\text{near}}, t_{\text{far}}] = [t_{\text{near}}, \text{split}]$ is not empty, and it visits the far node iff $[\text{split}, \infty] \cup [t_{\text{near}}, t_{\text{far}}] = [\text{split}, t_{\text{far}}]$ is not empty. This way, the interesting parameter interval of the ray is always available for the current node. The special cases, where the ray is parallel to the partitioning plane $(\text{split} = \infty)$ or pointing away from the partitioning plane $(\text{split} < 0)$, are easy to handle.

We described so far the traversal of a 2-D domain (the heightfield plane) by a kd-tree of dimension 2. For a 2.5-D heightfield, which is a real height function on the 2-D domain, the approach can be extended. If for each kd-tree node the maximum height is available, then it can be used to prune subtrees of the kd-tree from traversal. This simple extension requires only one additional real value per inner node of the kd-tree.

As the domain part represented by a kd-tree node shrinks with each subdivision, the leaf nodes represent single entries of the heightfield grid. From these discrete measurements, terrain surfaces can be reconstructed of various orders of continuity and of polynomial type. The question "which one represents the terrain best for a special application" has received considerable interest [21].

The height of the kd-tree for a heightfield of size $n \cdot m$ is $\log_2(n \cdot m)$, and the number of nodes is $2(n \cdot m))$. With overlapped splitting, the height is $\log_2(n \cdot m) + 2$ and $8(n \cdot m))$ nodes. The data per node consist of two pointers (4 bytes each on a 32-bit architecture), the real maximum height (8 bytes in double precision), and the real split value (8 bytes in double precision), so that the memory requirements sum up to 20 bytes per node. For comparison, the given heightfield array consists of $n \cdot m$ height values with 8 bytes per entry.

A. *Nonstandard Decomposition in Max-Plus Algebra*

The compressed grid for a heightfield of size $n \times m$ has the same memory requirements as the square grid of side length $\max(\tilde{n}, \tilde{m})$ resp. $2 \cdot \max(\tilde{n}, \tilde{m})$ with overlapped splitting, whereas $\tilde{x}$ denotes the smallest power of two greater than or equal to x.

The inefficiency for nonpower-of-two and nonsquare formats can be eliminated, for example, by tiling techniques (as done in JPEG2000 [22]) or by a Boolean sequence, which gives the orientation of the split, horizontal or vertical.

This way, the wavelet-like compressed grid is a different storage scheme for the previously presented kd-tree. However, if sufficient memory is available, the square power-of-two format with its implicit 4-ary splitting is beneficial for caching and runtime efficiency.

Using the notation introduced in [23] and [24], the heightfield decomposition can be described by

$$c_k^{j-1} = \max\left(c_{2k}^j, c_{2k+1}^j\right) \quad \text{and} \quad d_k^{j-1} = c_{2k}^j - c_{2k+1}^j$$

using maximum coefficients c_k^j and the corresponding detail coefficients d_k^j. The composition step is then calculated by

$$c_{2k}^{j+1} = c_k^j + \min\left(0, d_k^j\right) \quad \text{and} \quad c_{2k+1}^{j+1} = c_k^j - \max\left(0, d_k^j\right).$$

Fig. 3 illustrates the results of a nonstandard decomposition to a quadratic heightfield of size 220×220. In the nonstandard decomposition, the application of the filter alternates between columns and rows.

The analysis filter can be described in a short manner using operations in max-plus algebra $R_{\max}$ [25]. This way, the filter process is very similar to one of Haar wavelets, although the maximum filter does not satisfy the wavelet-filter definition within the algebra $R_{\max}$.

B. Nonstandard Decomposition in Real Algebra

In the previous section, we described a wavelet-like decomposition where the maximum value of a region is directly available in the representation. Here, we want to show how maximum computation is possible with a nonstandard wavelet decomposition in real algebra. This is possible using a linear combination of scalar basis functions on the domain. The nonstandard approach to generate 2-D basis functions $b(x, y)$ is to combine two 1-D basis functions in a tensor-product fashion, i.e.,

$$b(x, y) = b_1(x) \cdot b_2(y). \tag{1}$$

The 1-D basis functions $b(x)$ can be grouped into subspaces $b_{i,k} \in V_i$, which build a nested set of vector spaces $V_0 \subset V_1 \subset V_2 \subset \cdots$. As i increases, the dimension of V_i, and as a consequence thereof the power to represent a function, increases. The basis functions for the space V_i are known as scaling functions.

The orthogonal complement of V_j in V_{j+1} is called W_j and contains all functions in V_{j+1} that are orthogonal to all those in V_j according to a chosen inner product. The basis functions of W_j are called wavelets. Altogether, the space V_i is decomposed into

$$W_{i-1} \oplus V_{i-1} = W_{i-1} \oplus W_{i-2} \oplus \cdots \oplus W_0 \oplus V_0 \tag{2}$$

and for the 2-D tensor products $V_i V_i$

$$\begin{aligned} &W_{i-1}W_{i-1} \oplus W_{i-1}V_{i-1} \oplus V_{i-1}W_{i-1} \oplus V_{i-1}V_{i-1} \\ &\quad = W_{i-1}W_{i-1} \oplus W_{i-1}V_{i-1} \oplus V_{i-1}W_{i-1} \oplus \cdots V_0V_0. \end{aligned} \tag{3}$$

Furthermore, the spaces V_j and W_j contain also the translated functions $f(x - k2^{-j})$, k integer, with each function $f(x)$. Similarly, the scaled functions $f(2x)$ is in V_{j-1} for each function $f(x)$ in V_j. A function f in space V_iV_i can be represented as

$$f(x, y) = \sum_{(k,l)} c_{k,l} b_k(x) b_l(y) \tag{4}$$

where $\{b_k\}$ and $\{b_l\}$ span the spaces V_j and W_j with $0 \le j < i$. Due to the finite small support of basis functions $b_k(x)b_l(y)$, it is possible to express the maximum

$$\begin{aligned} &\max\{f(x, y) : (x, y) \in D_1 \times D_2\} \\ &\quad \le \sum_{(k,l)\in P(D_1,D_2)} c_{k,l} \max\{b_k|D_1\} \max\{b_l|D_2\} \\ &\qquad + \sum_{(k,l)\in N(D_1,D_2)} c_{k,l} \min\{b_k|D_1\} \min\{b_l|D_2\} \end{aligned} \tag{5}$$

as a linear combination of the basis function's minima/maxima, where $P(D_1, D_2) \cup N(D_1, D_2)$ are all index pairs, for which $support\{b_k\} \cap D_1 \ne \oslash$ and $support\{b_l\} \cap D_2 \ne \oslash$, and $P(D_1, D_2)$ are the index pairs with positive coefficients and $N(D_1, D_2)$ are the index pairs with negative coefficients, respectively. A sampled representation of $b_k \in V_j$ and $b_k \in W_j$ can be determined by reconstructing the basis function from an impulse. As the spaces V_j and W_j are highly structured, the minima/maxima of basis functions on their support can be precomputed from a prototype impulse and systematically tabulated. The sum (5) incorporates an increasing number of basis functions at higher levels of the decomposition, starting with the value corresponding to V_0V_0. At level j, there are $3 \cdot 2^j \cdot 2^j$ basis functions, and altogether, there are $1 + 3\sum_{j=0,\ldots,n} 2^j 2^j$ summands. For efficiently computing a lower/upper bound, we can precompute the maximum $cmax_j^1$ of all positive coefficients corresponding to space W_jV_j and the minimum $cmin_j^1$ of all negative coefficients corresponding to space W_jV_j at level j. Similarly, we denote the maximum $cmax_j^2$, the minimum $cmin_j^2$ corresponding to space W_jW_j, and $cmax_j^3$, $cmin_j^3$ corresponding to space V_jW_j. With these values, we can compute a weaker upper bound

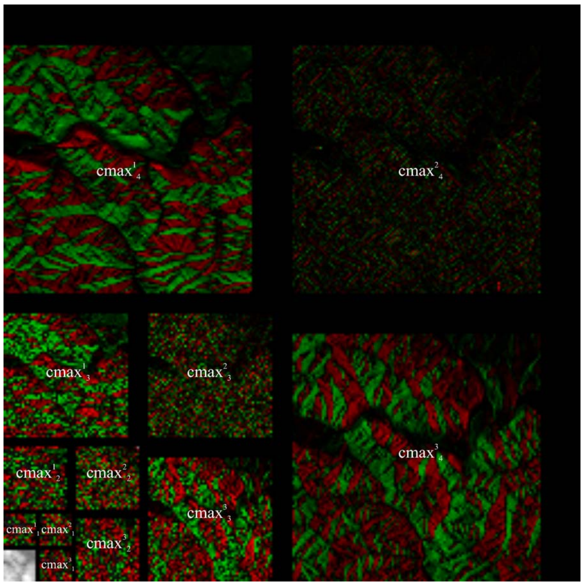


Fig. 4. Maximum values of positive coefficients (and, similarly, minimum values of negative coefficients) at different levels j.

$$\begin{aligned} &\max\{f(x, y) : (x, y) \in D_1 \times D_2\} \\ &\quad \le c_{0,0} \max\{b_0 | b_0 \in V_0V_0\} \\ &\qquad + \sum_{j=1,\ldots,n} cmax_j^1 \max\{b_1 | b_1 \in W_jV_j\} \\ &\qquad + \sum_{j=1,\ldots,n} cmin_j^1 \min\{b_1 | b_1 \in W_jV_j\} \\ &\qquad + \sum_{j=1,\ldots,n} cmax_j^2 \max\{b_2 | b_2 \in W_jW_j\} \\ &\qquad + \sum_{j=1,\ldots,n} cmin_j^2 \min\{b_2 | b_2 \in W_jW_j\} \\ &\qquad + \sum_{j=1,\ldots,n} cmax_j^3 \max\{b_3 | b_3 \in V_jW_j\} \\ &\qquad + \sum_{j=1,\ldots,n} cmin_j^3 \min\{b_3 | b_3 \in V_jW_j\} \end{aligned} \tag{6}$$

with only $6n + 1$ summands. Fig. 4 sketches the domain sections for which minimum/maximum values are precomputed.

TABLE I
TIMINGS OF LOS COMPUTATION ON THE SAME HEIGHTFIELDS IN DIFFERENT RESOLUTIONS (QUERIES TO TWO GROUND STATIONS PER HEIGHTFIELD PIXEL)

	220*220	256*256	512*512	1024*1024
kd-tree, point	0.00321ms 311463 $\frac{q}{s}$	0.00324ms 308214 $\frac{q}{s}$	0.00391ms 255696 $\frac{q}{s}$	0.00381ms 262440 $\frac{q}{s}$
kd-tree, bilinear-approx	0.00548ms 182468 $\frac{q}{s}$	0.00539ms 185671 $\frac{q}{s}$	0.00583ms 171488 $\frac{q}{s}$	0.00642ms 155867 $\frac{q}{s}$
kd-tree, bilinear	0.00600ms 178676 $\frac{q}{s}$	0.00557ms 179615 $\frac{q}{s}$	0.00558ms 179332 $\frac{q}{s}$	0.00697ms 143456 $\frac{q}{s}$
wavelet, point	0.00335ms 298808 $\frac{q}{s}$	0.00345ms 289764 $\frac{q}{s}$	0.00391ms 255636 $\frac{q}{s}$	0.00446ms 224383 $\frac{q}{s}$
wavelet, bilinear-approx	0.00553ms 180600 $\frac{q}{s}$	0.00568ms 175947 $\frac{q}{s}$	0.00597ms 167430 $\frac{q}{s}$	0.00698ms 143273 $\frac{q}{s}$
wavelet, bilinear	0.00546ms 183242 $\frac{q}{s}$	0.00581ms 172219 $\frac{q}{s}$	0.00605ms 165277 $\frac{q}{s}$	0.00713ms 140322 $\frac{q}{s}$

TABLE II
TIMINGS OF DIFFERENT SEARCH DIRECTIONS: FORWARD, BACKWARD, OR BASED ON WHICH ONE WAS FASTER IN THE LAST QUERY (HEIGHTFIELD OF RESOLUTION 220 × 220 WITH TWO GROUND STATIONS)

	forward	backward	last-fastest
kd-tree, point	0.00325ms 308084 $\frac{q}{s}$	0.00316ms 316837 $\frac{q}{s}$	0.00315ms 317766 $\frac{q}{s}$
kd-tree, bilinear	0.00612ms 163273 $\frac{q}{s}$	0.00556ms 179889 $\frac{q}{s}$	0.00563ms 177758 $\frac{q}{s}$
wavelet, point	0.00356ms 281024 $\frac{q}{s}$	0.00344ms 291051 $\frac{q}{s}$	0.00341ms 293648 $\frac{q}{s}$
wavelet, bilinear	0.00662ms 151095 $\frac{q}{s}$	0.00563ms 177482 $\frac{q}{s}$	0.00546ms 183242 $\frac{q}{s}$

C. Empirical Comparison

We have implemented the kd-tree data structure and its compressed variant with LOS computation. In this section, we give a short comparison of its performance in terms of data set sizes and search directions. A detailed comparison can be found in [26].

Table I lists the computation times for different resolutions of the same heightfield data set. All timings were taken on a Windows XP system with an Intel Pentium M 1.6-GHz processor and 1.5-GB RAM. Point reconstruction is fastest as it needs to access only a single height value at the leaf level. For bilinear approximate and bilinear reconstruction, we try to keep the four leaf nodes required for the reconstruction sequentially in memory. The resulting performance is roughly 3/4 of that of point reconstruction. In particular, it is notable that the wavelet-based storage scheme is nearly as fast as the kd-tree with fully stored inner nodes. For the more complex bilinear reconstruction in leaf nodes, the difference is even smaller.

The tests comparing search directions in Table II show that using the information where the intersection point was in the last query (at a different height or in a horizontal or vertical neighbor) is most successful. In addition, this information is very easy to exploit. Note that it is not easy to predict if a forward or backward search has a shorter search length up to an intersection as it requires carrying out the search.

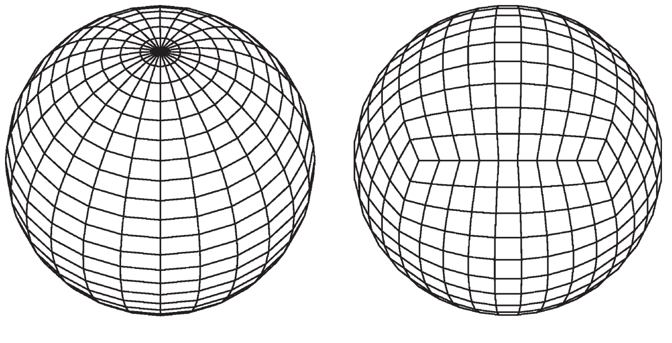


Fig. 5. Typical sphere parameterization and an alternative.

IV. COLLISION DETECTION BASED ON SPHERICAL DISTANCE FIELDS

Using a spherical distance field, we developed a collision detection algorithm [27]. In contrast to multiresolution mesh representations with wavelets [28] or progressive meshes [29], our approach is simpler and does not reproduce the model topology but allows one to efficiently generate conservative bounding volumes for model parts.

The spherical sampling according to a single center point requires a spherical parameterization, where we have chosen one with six charts derived from the box sides. The use of a spherical representation allows fast model rotation and on-the-fly generation of spherical shell bounding volumes for model parts.

Like most collision detection algorithms, this approach consists of two parts: a preprocessing step and a testing routine, which represents the intrinsic collision test.

During the preprocessing step, the algorithm takes an initial model and determines a model center. The choice of the center point and its position is very important for the sampling process. A center point, with respect to which the model is star shaped, is preferable. However, even if such a star exists, it is expensive to compute. Heuristic choices such as the center of an enclosing sphere or the model's mass center are good enough to serve as a sampling center. The result of the sampling process is a spherical heightfield $r(\phi, \theta)$ over the parameter domain $[0, 2\pi] \times [-(\pi/2), (\pi/2)]$, which encloses the whole object.

As in this parameterization all meridians coincide with each other at the poles, an intersection test in a near pole region would lead to many descending tests and a bad performance. Therefore, another sphere parameterization, which is illustrated in Fig. 5, is used. It subdivides a sphere into six separate congruent regions and applies an angle-based parameterization to each side. Subsampling of the six charts of our spherical representation is done analogous to a discretized Cartesian heightfield explained above. An illustrative example is shown in Fig. 6.

Having transformed all objects this way, it is possible to perform a simple and fast collision test. At runtime, the intersection test starts with the model representation at the lowest resolution and tests whether they collide or not. If this test is positive, the level of detail will be increased. Thereby, it is important for performance purposes that only intersecting sectors are considered further on.

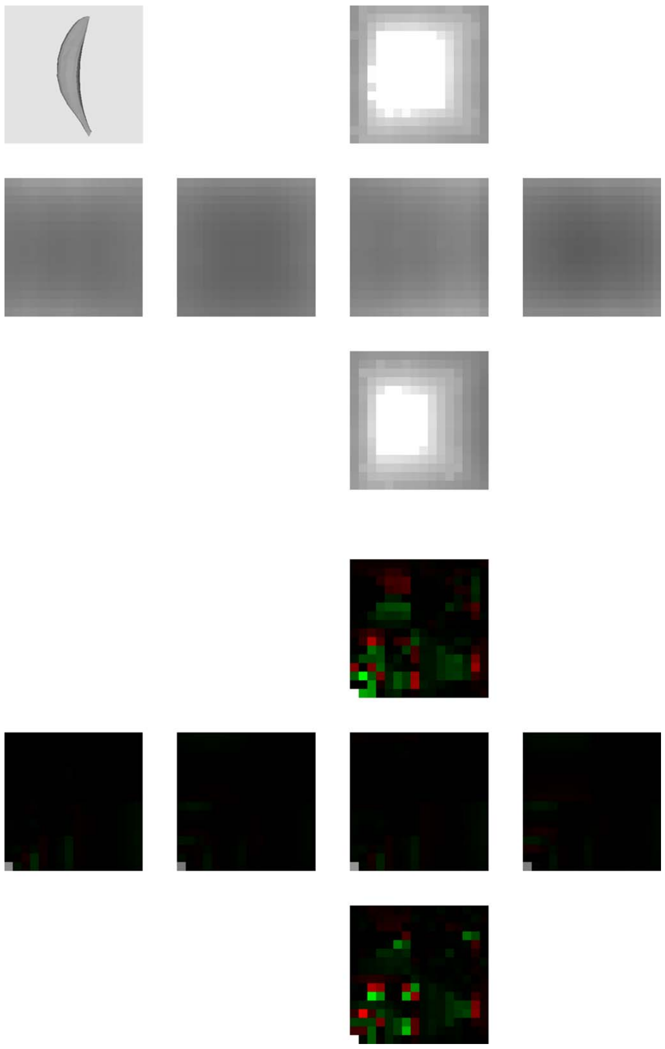


Fig. 6. Banana model, with (top) maximum distances on the six sides shown as grayscale images with a resolution of 16 × 16 and (bottom) corresponding transforms. The lower left sample contains the overall maximum distance. The other samples are differences as in a nonstandard decomposition, where red/green encodes the difference sign (negative/positive), and intensity encodes the difference magnitude (consequently, small differences are in black).

As long as there are intersecting sectors of different objects, the algorithm refines the objects. If the algorithm reaches the highest resolution of an object, the collision test is performed on the object primitives, which determines the colliding subparts.

The most relevant part concerning performance is the intersection test routine. Without a doubt, the test has to report an intersection, if there is one. However, if there is no intersection, the test may report one by mistake. The less faulty positive results are reported, the faster the algorithm works, as unnecessary refinements are omitted. Therefore, balancing accuracy and efficiency are essential.

At level 0, both objects to test are enclosed by tight spheres. The test, whether two spheres intersect each other, is that simple, that there is no need for simplification. However, at all other levels of detail, the algorithm has to check two sphere sections, which is analyzed in the list that follows.

1) So-called bounding volume tests enclose the objects to test within geometrically simpler objects; for example, a polygonal frustum might be used to enclose a sphere section. Although the intersection test of two polygonal frustums is rather simple, the amount of tests increases, and the enclosing quality is rather bad.
2) A totally different approach to check for intersections uses interval/affine arithmetics [30]. In theory, this approach offers an exact and fast intersection test. Although such a test might exist, a practical test has not yet been implemented.
3) The intersection test presented in [31] is based on algebraic tests for intersection. This test is suitable for our purposes, but due to its complexity and computational expense, we prefer a computationally cheaper alternative.

The following geometrical test shows reasonable performance. Checking two capped cones in 3-D is nontrivial. Subject to the orientation of the cone axes, two cases are analyzed. The first case deals with nonparallel axes, whereas the second case deals with parallel lines. A heuristically chosen (angle) threshold distinguishes between the nonparallel and parallel cases.

In the nonparallel case for both axes, the shortest distance and the corresponding perpendicular points P_1 and P_2 are determined. The test itself considers cone sections. One cone is intersected with a plane containing the first cone's center and the line through P_1 and P_2. This results in a well-known cone section. The first cone is reduced to a line, which passes the first cone's center and the point that you get by translating P_1 within the considered plane toward the cone section's focus. The intersection test is then reduced to a 2-D intersection test between a cone section and a line.

In the parallel case, the algorithm analyzes the projection of one cone onto the other cone's axis and vice versa. For each projection, two distance checks of points against their according radii are performed, which results in a total of four checks. The special case of parallel lines is separately handled for optimization, as in this case the whole test can be done by some interval checks, which are much faster.

Together with a quick rejection test (consider bounding cylinders instead of cones) and a quick acceptance test (consider spheres inside the cones) for the nonparallel case, the algorithm is reasonably fast.

Collision Detection Comparison: To show the efficiency of the new collision detection method, we have run a series of benchmarks on a Windows PC with a Pentium M 1.6-GHz processor. Realistically benchmarking collision detection algorithms is a difficult task. This is because there is a wealth of models with different characteristics and motions relative to each other. In [32], a benchmarking scheme for two models each contained in a unit box is proposed, where the second object performs a number of full-z rotations (in 1000 frames) at decreasing distances relative to the first object.

To compare this approach with well-known collision detection methods, we have included timings using an 18-sided discrete orientation polytope (18-DOP) [32] and oriented bounding boxes as in the publicly available library RAPID [33]. These approaches also report all triangle pairs in collision.

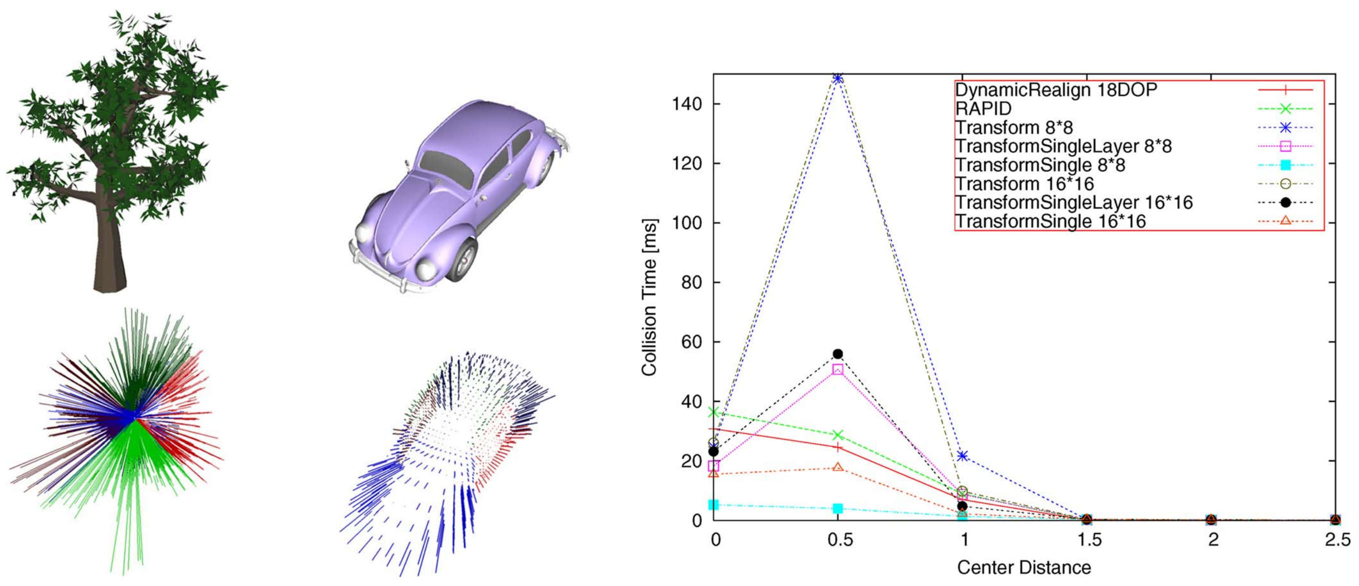


Fig. 7. Collision test for models Tree (4316 triangles) and Beetle (57 243 triangles). The line drawings show the axis of each capped cone of the discretized models. The benchmark "Transform" reports all triangle pairs in collision without elimination of duplicates; "TransformSingleLayer" reports a single triangle pair per layer of a colliding shell pair; and "TransformSingle" reports a single triangle pair in a colliding shell pair. For comparison, 18-DOP bounding volumes and oriented bounding volumes (public library RAPID) are shown.

The Fig. 7 contains the collision test models: a tree model colliding with a Beetle car model. This test demonstrates the algorithm's ability to handle all kinds of models. Absolutely unstructured polygon soups, as used for the leaves within the tree model, can be handled the same way like all other kinds of representations without any problems. Furthermore, the tree model is neither convex nor star shaped.

If reporting only a single triangle per spherical shell, the timings do not very much vary. In this mode of collision determination, the collision time is independent of the triangle count. It only depends on the sampling density and the volume between the inner and outer bounding shells.

V. Conclusion

We have presented a new collision detection algorithm, which is suitable for all model types, e.g., polygon soups, surfaces, and volumetric models. It is simple to implement, and its storage scheme consumes half the space compared to the full storage of the hierarchical spherical distance field.

The approach is scalable in the information it gives in collision determination. If it reports a single triangle per spherical shell, then the collision time only depends on the sampling density and the volume of spherical shells, which are used as bounding volumes, but not on the primitives' count of the model. Due to this fact, it is possible to tightly estimate the time bounds for the collision test. For bounding volume hierarchies, the worst-case time bounds are not tight in general, as the bounding volumes can arbitrarily overlap in space.

If we check all triangle pairs inside a spherical shell for intersection, then the approach works well in situations with few collisions, which are the most relevant in practice. In close-proximity situations, the algorithm degenerates to comparing triangle lists sorted according to the center distances of a triangle point.

Furthermore, we have shown that LOS calculation and collision detection can similarly be handled. Our wavelet-like storage scheme can be used with both arbitrary collision detection queries and LOS queries to create a more compact representation in memory.

References

[1] D. W. Fellner and N. Schenk, "MRT—A tool for simulations in 3D geometric domains," in *Proc. ESM*, Jun. 1997, pp. 185–188.

[2] A. Schmitz and M. Wenig, "The effect of the radio wave propagation model in mobile ad hoc networks," in *Proc. 9th ACM Int. Symp. Model., Anal. Simul. Wireless Mobile Syst. (MSWiM)*, 2006, pp. 61–67.

[3] M. Allegretti, M. Colaneri, R. Notarpietro, M. Gabella, and G. Perona, "Simulation in urban environment of a 3D ray tracing propagation model based on building database preprocessing," in *Proc. URSI Gen. Assem.*, 2005. [Online]. Available: www.ursi.org/Proceedings/ProcGA05/pdf/CP1.7(0958).pdf

[4] C. K. Yang, "Integration of volume visualization and compression: A survey," State Univ. New York, Stony Brook, NY, Tech. Rep. RPE-10, Aug. 2000.

[5] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross, "Optimized spatial hashing for collision detection of deformable objects," in *Proc. Vis., Model., Vis.*, Nov. 2003, pp. 47–54.

[6] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[7] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively sampled distance fields: A general representation of shape for computer graphics," in *Proc. SIGGRAPH*, 2000, pp. 249–254.

[8] S. Lefebvre and H. Hoppe, "Compressed random-access trees for spatially coherent data," in *Proc. EG Symp. Rendering*, Aug. 2007, pp. 339–350.

[9] M. D. Proctor and W. J. Gerber, "Line-of-sight attributes for a generalized application program interface," *JDMS*, vol. 1, no. 1, pp. 43–57, Apr. 2004.

[10] B. Salomon, N. Govindaraju, A. Sud, R. Gayle, M. Lin, D. Manocha, B. Butler, M. Bauer, A. Rodriguez, L. Eifert, A. Rubel, and M. Macedonia, "Accelerating line of sight computation using graphics processing units," in *Proc. 24th Army Sci. Conf.*, 2005. ADA433414. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA433414&Location=U2&doc=GetTRDoc.pdf

[11] D. Tuft, B. Salomon, S. Hanlon, and D. Manocha, "Fast line-of-sight computations in complex environments," Univ. North Carolina, Chapel Hill, NC, Tech. Rep. TR05-025, 2005.

[12] N. L. Max, "Horizon mapping: Shadows for bump-mapped surfaces," *Vis. Comput.*, vol. 4, no. 2, pp. 109–117, Mar. 1988.
[13] H. Rushmeier, L. Balmelli, and F. Bernardini, "Horizon map capture," *Comput. Graph. Forum*, vol. 20, no. 3, pp. 85–94, Sep. 2001.
[14] J. A. Stewart, "Fast horizon computation at all points of a terrain with visibility and shading applications," *IEEE Trans. Vis. Comput. Graph.*, vol. 4, no. 1, pp. 82–93, Jan. 1998.
[15] M. C. Lin and S. Gottschalk, "Collision detection between geometric models: A survey," in *Proc. 8th IMA Conf. Math. Surfaces*, 1998, pp. 37–56.
[16] M. C. Lin and D. Manocha, "Collision and proximity queries," in *Handbook of Discrete and Computational Geometry*. Boca Raton, FL: CRC Press, 2003.
[17] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino, "Collision detection for deformable objects," in *Proc. Eurographics*, 2004, pp. 119–140. State-of-the-Art Report.
[18] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six degrees-of-freedom haptic rendering using voxel sampling," in *Proc. SIGGRAPH*, 1999, pp. 401–408.
[19] A. Fuhrmann, G. Sobottka, and C. Groß, "Distance fields for rapid collision detection in physically based modeling," in *Proc. GraphiCon*, Sep. 2003, pp. 58–65.
[20] S. Fisher and M. C. Lin, "Deformed distance fields for simulation of non-penetrating flexible bodies," in *Proc. EG Workshop Comput. Animation Simul.*, 2001, pp. 99–111.
[21] D. Kidner, M. Dorey, and D. Smith, "What's the point? Interpolation and extrapolation with a regular grid DEM," in *Proc. GeoComputation*, 1999. [Online]. Available: http://www.geovista.psu.edu/sites/geocomp99/Gc99/082/gc_082.htm
[22] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding: An overview," *IEEE Trans. Consum. Electron.*, vol. 46, no. 4, pp. 1103–1127, Nov. 2000.
[23] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, "Wavelets for computer graphics: A primer, Part 1," *IEEE Comput. Graph. Appl.*, vol. 15, no. 3, pp. 76–84, May 1995.
[24] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, "Wavelets for computer graphics: A primer, Part 2," *IEEE Comput. Graph. Appl.*, vol. 15, no. 4, pp. 75–85, Jul. 1995.
[25] M. Akian, G. Cohen, S. Gaubert, R. Nikhoukhah, and J. P. Quadrat, "Linear systems in $(\max, +)$ algebra," in *Proc. 29th Conf. Decision Control*, Honolulu, HI, Dec. 1990, pp. 151–156.
[26] C. Fünfzig, T. Ullrich, D. W. Fellner, and E. N. Bachelder, "Empirical comparison of data structures for line-of-sight computation," in *Proc. IEEE Int. Symp. Intell. Signal Process. (WISP)*, 2007, vol. 1, pp. 291–296.
[27] C. Fünfzig, T. Ullrich, and D. W. Fellner, "Hierarchical spherical distance fields for collision detection," *IEEE Comput. Graph. Appl.*, vol. 26, no. 1, pp. 64–74, Jan./Feb. 2006.
[28] S. Valette and R. Prost, "Wavelet based multiresolution analysis of irregular surface meshes," *IEEE Trans. Vis. Comput. Graph.*, vol. 10, no. 2, pp. 113–122, Mar./Apr. 2004.
[29] H. Hoppe, "Efficient implementation of progressive meshes," *Comput. Graph.*, vol. 22, no. 1, pp. 27–36, Feb. 1998.
[30] K. Bühler, "Taylor models and affine arithmetics: Towards a more sophisticated use of reliable methods in computer graphics," in *Proc. Spring Conf. Comput. Graph.*, 2001, vol. 17, pp. 40–48.
[31] S. Krishnan, A. Pattekar, and M. C. Lin, "Spherical shell: A higher order bounding volume for fast proximity queries," in *Proc. 3rd Workshop Algorithmic Found. Robot.*, 1998, vol. 3, pp. 177–190.
[32] G. Zachmann, "Rapid collision detection by dynamically aligned DOP-trees," in *Proc. Virtual Reality Annu. Int. Symp.*, 1998, pp. 90–97.
[33] S. Gottschalk, M. C. Lin, and D. Manocha, "OBB-tree: A hierarchical structure for rapid interference detection," in *Proc. SIGGRAPH*, 1996, pp. 171–180.

Christoph Fünfzig received the M.Sc. degree from the University Bonn, Bonn, Germany, and the Ph.D. degree in computer science from Braunschweig University of Technology, Braunschweig, Germany.

He is currently with the CAGD Group, Univ. Valenciennes et du Hainaut-Cambrésis, FR CNRS 2956, Valenciennes, France. His research interests include practical computational geometry, animation and simulation in computer graphics, virtual/augmented reality, and scientific visualization.

Torsten Ullrich received the M.Sc. degree in mathematics from Karlsruhe Institute of Technology, Karlsruhe, Germany. He is currently working toward the Ph.D. degree in computer science with Graz University of Technology, Graz, Austria.

His research has been concerned with computer-aided geometric design topics, including modeling and reconstruction.

Dieter W. Fellner received the M.Sc. and Ph.D. degrees from Graz University of Technology, Graz, Austria.

He is the Director of the Fraunhofer Institute of Computer Graphics (IGD), Darmstadt, Germany, and a Professor of computer science with Darmstadt University of Technology (TU Darmstadt), Darmstadt, with a joint affiliation with Graz University of Technology. His research interests include computer graphics, modeling, immersive systems, and graphics in digital libraries. He has held academic positions with universities in Graz, Austria; Denver, CO; St. John's, NF, Canada; Bonn, Germany; and Braunschweig, Germany.

Dr. Fellner is a member of the Association for Computing Machinery, Eurographics, and Gesellschaft für Informatik (GI). He is on the Editorial Board of several international journals, IEEE COMPUTER GRAPHICS AND APPLICATIONS being one of them.

Edward N. Bachelder received the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge.

Prior to receiving his Ph.D. degree, he was a Naval Aviator flying the SH-60B. He is currently a Principal Research engineer with Systems Technology Inc., Hawthorne, CA. His areas of research include augmented reality, optimized control guidance for helicopter autorotation training and operation, real-time path optimization, system identification (extremely low to very high frequency regimes) using sparse excitation, 3-D helicopter cueing for precision hover, and nap-of-earth flight.

Terrain and Model Queries Using Scalar Representations With Wavelet Compression

Christoph Fünfzig, Torsten Ullrich, Dieter W. Fellner, and Edward N. Bachelder

***Abstract*—In this paper, we present efficient height/distance field data structures for line-of-sight (LOS) queries on terrains and collision queries on arbitrary 3-D models. The data structure uses a pyramid of quad-shaped regions with the original height/distance field at the highest level and an overall minimum/maximum value at the lower levels. The pyramid can compactly be stored in a wavelet-like decomposition but using max and plus operations. Additionally, we show how to get minimum/maximum values for regions in a wavelet decomposition using real algebra. For LOS calculations, we compare with a kd-tree representation containing the maximum height values. Furthermore, we show that the LOS calculation is a special case of a collision detection query. Using our wavelet-like approach, even general and arbitrary collision detection queries can efficiently be answered.**

***Index Terms*—Collision detection, distance fields, heightfield interrogation, kd-tree data structure, line-of-sight (LOS) computation, pyramid algorithms, wavelets.**

I. Introduction

THE SIMULATION and planning of realistic movements governed by physics is necessary for many applications. The simulation of movements is often based on collision detection, as changes of movements occur at events, where geometries collide.

Planning solutions like signal strength prediction [1], [2] and antenna placement [3] may use ray-casting models for electromagnetic wave propagation. In these models, the main propagation paths are evaluated by rays (geometric optics), and special wave effects are added at the intersection points of the rays with the terrain surface. Of course, the line-of-sights (LOSs) from the antennas account for the main propagation paths. Thus, efficient LOS computation is an important means for terrain interrogation.

In this paper, we present efficient height/distance field data structures for LOS queries on terrains and collision queries on arbitrary 3-D models. The data structure uses a pyramid of quad-shaped regions with the original height/distance field at the highest level and maximum values of regions at the lower levels (*maximum mipmap heightfield*). The pyramid can compactly be stored in a wavelet-like decomposition but using max and plus operations. Additionally, we show how to get minimum/maximum values for regions in a nonstandard wavelet decomposition in real algebra (*wavelet heightfield*). For heightfields (like regular-grid terrains), the data structure is similar to a 2-D kd-tree with added minimum/maximum height values in inner nodes.

The technique for a single heightfield can be extended to a spherical distance field for arbitrary 3-D models. With the spherical distance field according to a fixed center point, conservative distance bounds to the model can efficiently be queried. Furthermore, we show that even general and arbitrary collision detection queries can similarly be answered to LOS queries.

Applications: Height/distance field data sets are required in various applications. Due to their size, they are commonly compressed using a wavelet basis. Fig. 1 shows an example of a distance data set, compressed with the Daubechies-4 wavelet and its subband coefficients adaptively quantized. The motivation for this work is to directly use the compressed data sets (without a complete decompression step) for queries (collision and LOS tests). We call this *on-the-fly decompression during querying*, which is similar to the coupling of decompression with rendering [4]. The benefits are much lower memory requirements at the same or even smaller runtime. The problem of how to get minimum/maximum values for representations with wavelets in real algebra is covered in Section III-B. Using a special Haar wavelet-like decomposition in max-plus algebra, the maximum values of quad-shaped subregions are directly available from the representation. Due to their small support and discontinuity, they are worse in terms of compression efficiency but are very fast to reconstruct and query. The construction and reconstruction of the maximum mipmap heightfields are described in Section III-A.

II. Related Work

Basic data structures used for the LOS computation problem are the grid structure and several tree-structured schemes. The grid structure is a sampled representation, directly storing height/distance values to a reference plane or reference surface. There have been several approaches of compressing the grid data, among them spatial hashing [5] and wavelet transform [4]. The kd-tree is a binary tree with splits cycling through the d dimensions. It has been invented for organizing point sets for

Manuscript received January 30, 2008; revised June 29, 2008. The Associate Editor coordinating the review process for this paper was Dr. Annamaria Varkonyi-Koczy.

C. Fünfzig is with the CAGD Group, Univ. Valenciennes et du Hainaut-Cambrésis, FR CNRS 2956, 59300 Valenciennes, France (e-mail: c.fuenfzig@gmx.de).

T. Ullrich is with the Institute of Computer Graphics and Knowledge Visualization, Graz University of Technology, 8010 Graz, Austria (e-mail: t.ullrich@cgv.tugraz.at).

D. W. Fellner is with the Department of Graphisch-Interaktive Systeme (GRIS), Darmstadt University of Technology (TU Darmstadt), 64283 Darmstadt, Germany (e-mail: d.fellner@igd.fhg.de).

E. N. Bachelder is with Systems Technology Inc., Hawthorne, CA 90250 USA (e-mail: edbach@systemstech.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIM.2009.2016879

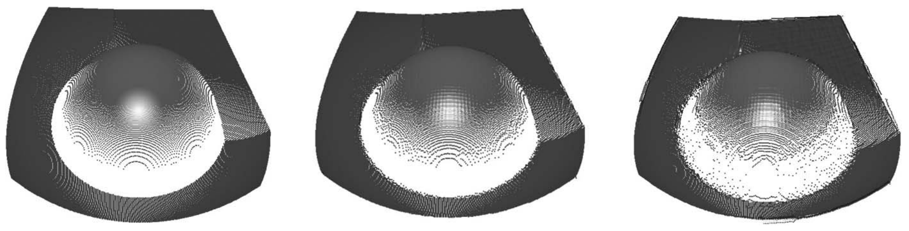


Fig. 1. Reconstruction of point cloud (256×256) from Daubechies-4 wavelet representations. The scene consists of a sphere in the center and a plane in the back right. (Left to right) 32-bit quantization in all subbands ($\text{MSE} = 1.92876e-019; \text{MAXSE} = 3.78111e-018$); 32 bits in bands 0–3, 16 bits in bands 4 and 5, 8 bits in band 6, and 4 bits in band 7 ($\text{MSE} = 8.36761e-005; \text{MAXSE} = 0.0261943$); and 32 bits in bands 0 and 1, 16 bits in bands 2 and 3, 8 bits in bands 4 and 5, 4 bits in band 6, and 2 bits in band 7 ($\text{MSE} = 0.00229742; \text{MAXSE} = 0.78619$). The MSE is the mean square error, i.e., the mean of square differences between the original and the reconstructed distance values over the full image. The MAXSE is the maximum of all square difference values.

nearest neighbor searching [6] and stores data for rectangular spatial cells in its nodes. Frisken *et al.* [7] use simultaneous splits of all dimensions resulting in a 2^d-ary tree and store data for the cell corners. With bilinear interpolation on the space region, they represent a continuous scalar function. Lefebvre and Hoppe [8] covers the problem of encoding the tree topology and the tree data with a combination of techniques suitable for random access.

LOS Calculation: The most straightforward approach to calculate LOS traverses the projection of the ray on the domain plane and checks the ray heights against the heightfield heights at a number of points per cell [9]. Different terrain reconstruction is possible for noninteger grid positions: point reconstruction, double linear interpolation, and bilinear interpolation. Without a specific test for the reconstruction used, an exact LOS test is not possible. By checking a number p of discrete points per grid cell, large low-height regions cannot be exploited. The approach always requires $l \cdot p$ height evaluations regardless of the terrain traversed, where l is the ray length in cells.

Recently, graphics processing units (GPU) have also been used to determine the LOS on a terrain (see [10] and [11]). These approaches render both the terrain surface and the ray line and test if all line fragments are above the terrain surface. If this is the case, it is a LOS up to the image resolution used for rendering and for terrain reconstruction. A special hardware occlusion test is used for counting the number of visible fragments.

There is also work on slightly extended visibility problems like computing the horizon for each grid point and direction sector [12]–[14]. Originally, the resulting horizon map was invented for self-shadowing the terrain surface.

Collision Detection: Naive approaches to determine all collisions of n object parts require a runtime of $O(n^2)$, which will rapidly be too much in practice. Innumerable algorithms with reduced runtime complexity have been developed in the last decades. They can be classified according to different schemes with regard to, e.g., field of application or solution strategy.

A general approach for collision detection uses hierarchies of simple bounding volumes containing model parts. Researchers have proposed several bounding volumes, including spheres, axis-aligned bounding boxes, oriented bounding boxes, and discrete orientation polytopes. Here, the performance depends on the tightness of the bounding volume, the efficiency of the intersection test for the bounding volume, and the strategy for hierarchy generation [15], [16].

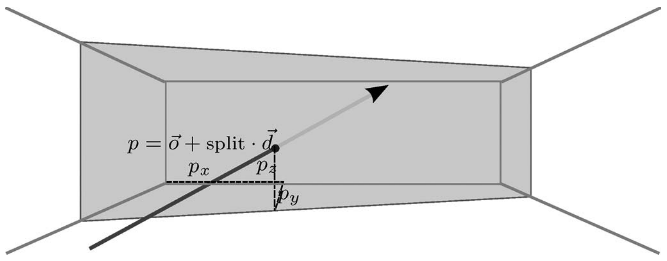


Fig. 2. Partitioning plane subdividing the ray into a near segment and a far segment at the ray–plane intersection point.

Bounding volume hierarchies with deformable models require additional time for refitting. Several papers discuss optimizations for special deformations like linear morphing, linear blend skinning, and linear combined displacement fields [17]. In this area, simpler ones like Cartesian grids and 1-D arrays accessed by hashing are commonly used. McNeely *et al.* [18] and Teschner *et al.* [5] build a voxel grid for the static model, where the points of a small movable model are queried against. This approach guarantees the high feedback rate needed by a haptic feedback device, and it is tailored to the haptic application domain.

Several authors [19], [20] use Cartesian distance fields for the implicit representation of models.

III. LOS Calculation Based on KD-Trees

A ray is defined by an origin point and a direction vector, which implicitly gives a search order on the data domain traversed by the ray. Thus, if the data domain is partitioned into several parts, then these parts can be searched according to the ray. *LOS computation* is a problem of this kind. Additionally, the parameter interval of the ray inside a domain part is available during traversal.

The simplest partition tree for a 2-D domain is a binary tree with domain-orthogonal partitioning planes. The parameter interval $[0, \infty]$ is subdivided by the ray–plane intersection point at parameter $\text{split} = \vec{n} \cdot (\vec{p} - \vec{o}) / \vec{n} \cdot \vec{d}$ into the near interval $[0, \text{split}]$ and the far interval $[\text{split}, \infty]$ (Fig. 2).

For axis-orthogonal partitioning planes, the ray–plane intersection computation $\text{split} = (p_x - o_x)/d_x$ resp. $\text{split} = (p_y - o_y)/d_y$ is simple and, therefore, particularly fast to compute. The resulting partitioning tree with alternating x- and y-orthogonal planes is called a *kd-tree* of dimension 2. The

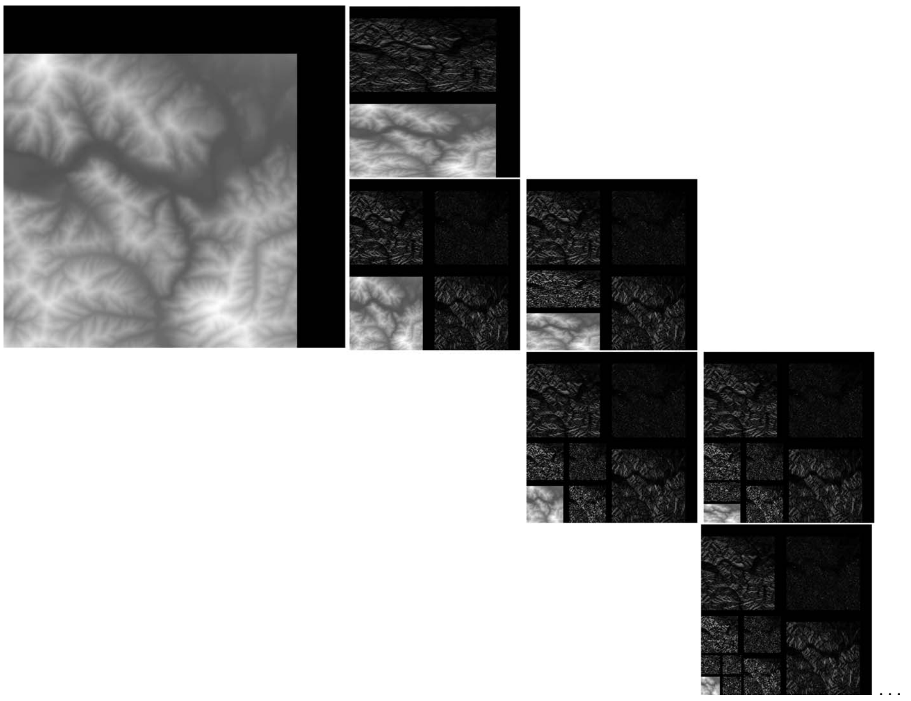


Fig. 3. (Left) Two-dimensional heightfield shown as a grayscale image. Its nonstandard decomposition using a maximum operation. Green pixels mark positive detail coefficients d_k^j, and red pixels mark negative coefficients d_k^j. A pixel's intensity encodes the difference magnitude (consequently, small differences are in black).

traversal of the kd-tree can be adapted for ray segments restricted to a parameter interval $[t_{\text{near}}, t_{\text{far}}]$. The traversal visits the near node iff $[0, \text{split}] \cup [t_{\text{near}}, t_{\text{far}}] = [t_{\text{near}}, \text{split}]$ is not empty, and it visits the far node iff $[\text{split}, \infty] \cup [t_{\text{near}}, t_{\text{far}}] = [\text{split}, t_{\text{far}}]$ is not empty. This way, the interesting parameter interval of the ray is always available for the current node. The special cases, where the ray is parallel to the partitioning plane $(\text{split} = \infty)$ or pointing away from the partitioning plane $(\text{split} < 0)$, are easy to handle.

We described so far the traversal of a 2-D domain (the heightfield plane) by a kd-tree of dimension 2. For a 2.5-D heightfield, which is a real height function on the 2-D domain, the approach can be extended. If for each kd-tree node the maximum height is available, then it can be used to prune subtrees of the kd-tree from traversal. This simple extension requires only one additional real value per inner node of the kd-tree.

As the domain part represented by a kd-tree node shrinks with each subdivision, the leaf nodes represent single entries of the heightfield grid. From these discrete measurements, terrain surfaces can be reconstructed of various orders of continuity and of polynomial type. The question "which one represents the terrain best for a special application" has received considerable interest [21].

The height of the kd-tree for a heightfield of size $n \cdot m$ is $\log_2(n \cdot m)$, and the number of nodes is $2(n \cdot m))$. With overlapped splitting, the height is $\log_2(n \cdot m) + 2$ and $8(n \cdot m))$ nodes. The data per node consist of two pointers (4 bytes each on a 32-bit architecture), the real maximum height (8 bytes in double precision), and the real split value (8 bytes in double precision), so that the memory requirements sum up to 20 bytes per node. For comparison, the given heightfield array consists of $n \cdot m$ height values with 8 bytes per entry.

A. *Nonstandard Decomposition in Max-Plus Algebra*

The compressed grid for a heightfield of size $n \times m$ has the same memory requirements as the square grid of side length $\max(\tilde{n}, \tilde{m})$ resp. $2 \cdot \max(\tilde{n}, \tilde{m})$ with overlapped splitting, whereas $\tilde{x}$ denotes the smallest power of two greater than or equal to x.

The inefficiency for nonpower-of-two and nonsquare formats can be eliminated, for example, by tiling techniques (as done in JPEG2000 [22]) or by a Boolean sequence, which gives the orientation of the split, horizontal or vertical.

This way, the wavelet-like compressed grid is a different storage scheme for the previously presented kd-tree. However, if sufficient memory is available, the square power-of-two format with its implicit 4-ary splitting is beneficial for caching and runtime efficiency.

Using the notation introduced in [23] and [24], the heightfield decomposition can be described by

$$c_k^{j-1} = \max\left(c_{2k}^j, c_{2k+1}^j\right) \quad \text{and} \quad d_k^{j-1} = c_{2k}^j - c_{2k+1}^j$$

using maximum coefficients c_k^j and the corresponding detail coefficients d_k^j. The composition step is then calculated by

$$c_{2k}^{j+1} = c_k^j + \min\left(0, d_k^j\right) \quad \text{and} \quad c_{2k+1}^{j+1} = c_k^j - \max\left(0, d_k^j\right).$$

Fig. 3 illustrates the results of a nonstandard decomposition to a quadratic heightfield of size 220×220. In the nonstandard decomposition, the application of the filter alternates between columns and rows.

The analysis filter can be described in a short manner using operations in max-plus algebra $R_{\max}$ [25]. This way, the filter process is very similar to one of Haar wavelets, although the maximum filter does not satisfy the wavelet-filter definition within the algebra $R_{\max}$.

B. Nonstandard Decomposition in Real Algebra

In the previous section, we described a wavelet-like decomposition where the maximum value of a region is directly available in the representation. Here, we want to show how maximum computation is possible with a nonstandard wavelet decomposition in real algebra. This is possible using a linear combination of scalar basis functions on the domain. The nonstandard approach to generate 2-D basis functions $b(x, y)$ is to combine two 1-D basis functions in a tensor-product fashion, i.e.,

$$b(x, y) = b_1(x) \cdot b_2(y). \quad (1)$$

The 1-D basis functions $b(x)$ can be grouped into subspaces $b_{i,k} \in V_i$, which build a nested set of vector spaces $V_0 \subset V_1 \subset V_2 \subset \cdots$. As i increases, the dimension of V_i, and as a consequence thereof the power to represent a function, increases. The basis functions for the space V_i are known as scaling functions.

The orthogonal complement of V_j in V_{j+1} is called W_j and contains all functions in V_{j+1} that are orthogonal to all those in V_j according to a chosen inner product. The basis functions of W_j are called wavelets. Altogether, the space V_i is decomposed into

$$W_{i-1} \oplus V_{i-1} = W_{i-1} \oplus W_{i-2} \oplus \cdots \oplus W_0 \oplus V_0 \quad (2)$$

and for the 2-D tensor products $V_i V_i$

$$\begin{aligned} &W_{i-1}W_{i-1} \oplus W_{i-1}V_{i-1} \oplus V_{i-1}W_{i-1} \oplus V_{i-1}V_{i-1} \\ &\quad = W_{i-1}W_{i-1} \oplus W_{i-1}V_{i-1} \oplus V_{i-1}W_{i-1} \oplus \cdots V_0V_0. \quad (3) \end{aligned}$$

Furthermore, the spaces V_j and W_j contain also the translated functions $f(x - k2^{-j})$, k integer, with each function $f(x)$. Similarly, the scaled functions $f(2x)$ is in V_{j-1} for each function $f(x)$ in V_j. A function f in space $V_i V_i$ can be represented as

$$f(x, y) = \sum_{(k,l)} c_{k,l} b_k(x) b_l(y) \quad (4)$$

where $\{b_k\}$ and $\{b_l\}$ span the spaces V_j and W_j with $0 \le j < i$. Due to the finite small support of basis functions $b_k(x)b_l(y)$, it is possible to express the maximum

$$\begin{aligned} &\max\{f(x, y) : (x, y) \in D_1 \times D_2\} \\ &\quad \le \sum_{(k,l) \in P(D_1, D_2)} c_{k,l} \max\{b_k | D_1\} \max\{b_l | D_2\} \\ &\qquad + \sum_{(k,l) \in N(D_1, D_2)} c_{k,l} \min\{b_k | D_1\} \min\{b_l | D_2\} \quad (5) \end{aligned}$$

as a linear combination of the basis function's minima/maxima, where $P(D_1, D_2) \cup N(D_1, D_2)$ are all index pairs, for which $support\{b_k\} \cap D_1 \ne \oslash$ and $support\{b_l\} \cap D_2 \ne \oslash$, and $P(D_1, D_2)$ are the index pairs with positive coefficients and $N(D_1, D_2)$ are the index pairs with negative coefficients, respectively. A sampled representation of $b_k \in V_j$ and $b_k \in W_j$ can be determined by reconstructing the basis function from an impulse. As the spaces V_j and W_j are highly structured, the minima/maxima of basis functions on their support can be precomputed from a prototype impulse and systematically tabulated. The sum (5) incorporates an increasing number of basis functions at higher levels of the decomposition, starting with the value corresponding to $V_0 V_0$. At level j, there are $3 \cdot 2^j \cdot 2^j$ basis functions, and altogether, there are $1 + 3 \sum_{j=0,\ldots,n} 2^j 2^j$ summands. For efficiently computing a lower/upper bound, we can precompute the maximum $cmax_j^1$ of all positive coefficients corresponding to space $W_j V_j$ and the minimum $cmin_j^1$ of all negative coefficients corresponding to space $W_j V_j$ at level j. Similarly, we denote the maximum $cmax_j^2$, the minimum $cmin_j^2$ corresponding to space $W_j W_j$, and $cmax_j^3$, $cmin_j^3$ corresponding to space $V_j W_j$. With these values, we can compute a weaker upper bound

$$\begin{aligned} &\max\{f(x, y) : (x, y) \in D_1 \times D_2\} \\ &\quad \le c_{0,0} \max\{b_0 | b_0 \in V_0 V_0\} \\ &\qquad + \sum_{j=1,\ldots,n} cmax_j^1 \max\{b_1 | b_1 \in W_j V_j\} \\ &\qquad + \sum_{j=1,\ldots,n} cmin_j^1 \min\{b_1 | b_1 \in W_j V_j\} \\ &\qquad + \sum_{j=1,\ldots,n} cmax_j^2 \max\{b_2 | b_2 \in W_j W_j\} \\ &\qquad + \sum_{j=1,\ldots,n} cmin_j^2 \min\{b_2 | b_2 \in W_j W_j\} \\ &\qquad + \sum_{j=1,\ldots,n} cmax_j^3 \max\{b_3 | b_3 \in V_j W_j\} \\ &\qquad + \sum_{j=1,\ldots,n} cmin_j^3 \min\{b_3 | b_3 \in V_j W_j\} \quad (6) \end{aligned}$$

with only $6n + 1$ summands. Fig. 4 sketches the domain sections for which minimum/maximum values are precomputed.

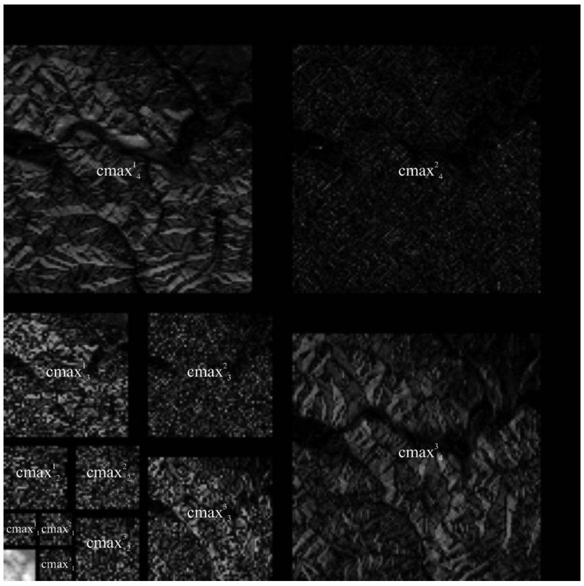


Fig. 4. Maximum values of positive coefficients (and, similarly, minimum values of negative coefficients) at different levels j.

TABLE I
TIMINGS OF LOS COMPUTATION ON THE SAME HEIGHTFIELDS IN DIFFERENT RESOLUTIONS (QUERIES TO TWO GROUND STATIONS PER HEIGHTFIELD PIXEL)

	220*220	256*256	512*512	1024*1024
kd-tree, point	0.00321ms 311463 $\frac{q}{s}$	0.00324ms 308214 $\frac{q}{s}$	0.00391ms 255696 $\frac{q}{s}$	0.00381ms 262440 $\frac{q}{s}$
kd-tree, bilinear-approx	0.00548ms 182468 $\frac{q}{s}$	0.00539ms 185671 $\frac{q}{s}$	0.00583ms 171488 $\frac{q}{s}$	0.00642ms 155867 $\frac{q}{s}$
kd-tree, bilinear	0.00600ms 178676 $\frac{q}{s}$	0.00557ms 179615 $\frac{q}{s}$	0.00558ms 179332 $\frac{q}{s}$	0.00697ms 143456 $\frac{q}{s}$
wavelet, point	0.00335ms 298808 $\frac{q}{s}$	0.00345ms 289764 $\frac{q}{s}$	0.00391ms 255636 $\frac{q}{s}$	0.00446ms 224383 $\frac{q}{s}$
wavelet, bilinear-approx	0.00553ms 180600 $\frac{q}{s}$	0.00568ms 175947 $\frac{q}{s}$	0.00597ms 167430 $\frac{q}{s}$	0.00698ms 143273 $\frac{q}{s}$
wavelet, bilinear	0.00546ms 183242 $\frac{q}{s}$	0.00581ms 172219 $\frac{q}{s}$	0.00605ms 165277 $\frac{q}{s}$	0.00713ms 140322 $\frac{q}{s}$

TABLE II
TIMINGS OF DIFFERENT SEARCH DIRECTIONS: FORWARD, BACKWARD, OR BASED ON WHICH ONE WAS FASTER IN THE LAST QUERY (HEIGHTFIELD OF RESOLUTION 220 × 220 WITH TWO GROUND STATIONS)

	forward	backward	last-fastest
kd-tree, point	0.00325ms 308084 $\frac{q}{s}$	0.00316ms 316837 $\frac{q}{s}$	0.00315ms 317766 $\frac{q}{s}$
kd-tree, bilinear	0.00612ms 163273 $\frac{q}{s}$	0.00556ms 179889 $\frac{q}{s}$	0.00563ms 177758 $\frac{q}{s}$
wavelet, point	0.00356ms 281024 $\frac{q}{s}$	0.00344ms 291051 $\frac{q}{s}$	0.00341ms 293648 $\frac{q}{s}$
wavelet, bilinear	0.00662ms 151095 $\frac{q}{s}$	0.00563ms 177482 $\frac{q}{s}$	0.00546ms 183242 $\frac{q}{s}$

C. Empirical Comparison

We have implemented the kd-tree data structure and its compressed variant with LOS computation. In this section, we give a short comparison of its performance in terms of data set sizes and search directions. A detailed comparison can be found in [26].

Table I lists the computation times for different resolutions of the same heightfield data set. All timings were taken on a Windows XP system with an Intel Pentium M 1.6-GHz processor and 1.5-GB RAM. Point reconstruction is fastest as it needs to access only a single height value at the leaf level. For bilinear approximate and bilinear reconstruction, we try to keep the four leaf nodes required for the reconstruction sequentially in memory. The resulting performance is roughly 3/4 of that of point reconstruction. In particular, it is notable that the wavelet-based storage scheme is nearly as fast as the kd-tree with fully stored inner nodes. For the more complex bilinear reconstruction in leaf nodes, the difference is even smaller.

The tests comparing search directions in Table II show that using the information where the intersection point was in the last query (at a different height or in a horizontal or vertical neighbor) is most successful. In addition, this information is very easy to exploit. Note that it is not easy to predict if a forward or backward search has a shorter search length up to an intersection as it requires carrying out the search.

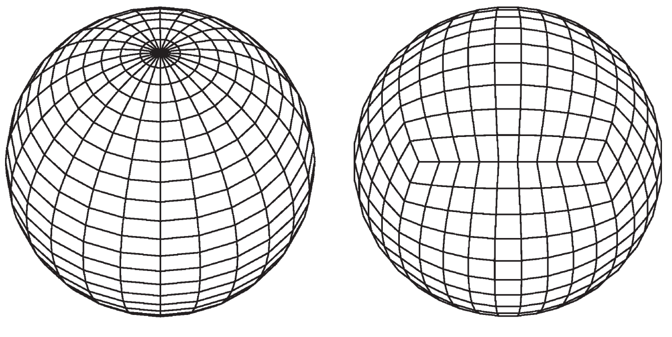


Fig. 5. Typical sphere parameterization and an alternative.

IV. COLLISION DETECTION BASED ON SPHERICAL DISTANCE FIELDS

Using a spherical distance field, we developed a collision detection algorithm [27]. In contrast to multiresolution mesh representations with wavelets [28] or progressive meshes [29], our approach is simpler and does not reproduce the model topology but allows one to efficiently generate conservative bounding volumes for model parts.

The spherical sampling according to a single center point requires a spherical parameterization, where we have chosen one with six charts derived from the box sides. The use of a spherical representation allows fast model rotation and on-the-fly generation of spherical shell bounding volumes for model parts.

Like most collision detection algorithms, this approach consists of two parts: a preprocessing step and a testing routine, which represents the intrinsic collision test.

During the preprocessing step, the algorithm takes an initial model and determines a model center. The choice of the center point and its position is very important for the sampling process. A center point, with respect to which the model is star shaped, is preferable. However, even if such a star exists, it is expensive to compute. Heuristic choices such as the center of an enclosing sphere or the model's mass center are good enough to serve as a sampling center. The result of the sampling process is a spherical heightfield $r(\phi, \theta)$ over the parameter domain $[0, 2\pi] \times [-(\pi/2), (\pi/2)]$, which encloses the whole object.

As in this parameterization all meridians coincide with each other at the poles, an intersection test in a near pole region would lead to many descending tests and a bad performance. Therefore, another sphere parameterization, which is illustrated in Fig. 5, is used. It subdivides a sphere into six separate congruent regions and applies an angle-based parameterization to each side. Subsampling of the six charts of our spherical representation is done analogous to a discretized Cartesian heightfield explained above. An illustrative example is shown in Fig. 6.

Having transformed all objects this way, it is possible to perform a simple and fast collision test. At runtime, the intersection test starts with the model representation at the lowest resolution and tests whether they collide or not. If this test is positive, the level of detail will be increased. Thereby, it is important for performance purposes that only intersecting sectors are considered further on.

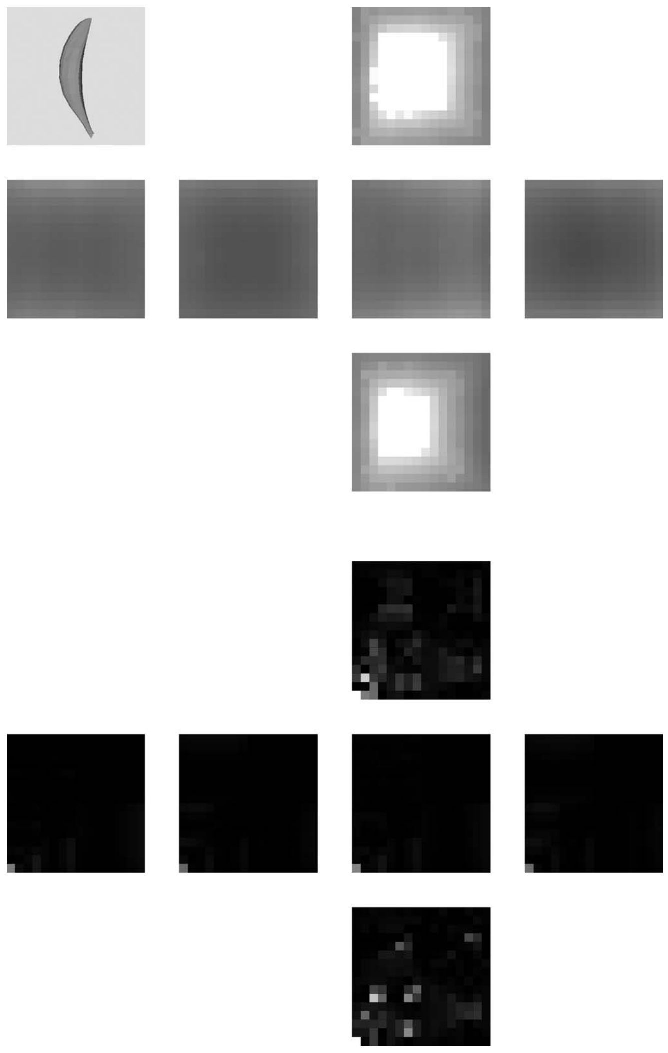


Fig. 6. Banana model, with (top) maximum distances on the six sides shown as grayscale images with a resolution of 16×16 and (bottom) corresponding transforms. The lower left sample contains the overall maximum distance. The other samples are differences as in a nonstandard decomposition, where red/green encodes the difference sign (negative/positive), and intensity encodes the difference magnitude (consequently, small differences are in black).

As long as there are intersecting sectors of different objects, the algorithm refines the objects. If the algorithm reaches the highest resolution of an object, the collision test is performed on the object primitives, which determines the colliding subparts.

The most relevant part concerning performance is the intersection test routine. Without a doubt, the test has to report an intersection, if there is one. However, if there is no intersection, the test may report one by mistake. The less faulty positive results are reported, the faster the algorithm works, as unnecessary refinements are omitted. Therefore, balancing accuracy and efficiency are essential.

At level 0, both objects to test are enclosed by tight spheres. The test, whether two spheres intersect each other, is that simple, that there is no need for simplification. However, at all other levels of detail, the algorithm has to check two sphere sections, which is analyzed in the list that follows.

1) So-called bounding volume tests enclose the objects to test within geometrically simpler objects; for example, a polygonal frustum might be used to enclose a sphere section. Although the intersection test of two polygonal frustums is rather simple, the amount of tests increases, and the enclosing quality is rather bad.
2) A totally different approach to check for intersections uses interval/affine arithmetics [30]. In theory, this approach offers an exact and fast intersection test. Although such a test might exist, a practical test has not yet been implemented.
3) The intersection test presented in [31] is based on algebraic tests for intersection. This test is suitable for our purposes, but due to its complexity and computational expense, we prefer a computationally cheaper alternative.

The following geometrical test shows reasonable performance. Checking two capped cones in 3-D is nontrivial. Subject to the orientation of the cone axes, two cases are analyzed. The first case deals with nonparallel axes, whereas the second case deals with parallel lines. A heuristically chosen (angle) threshold distinguishes between the nonparallel and parallel cases.

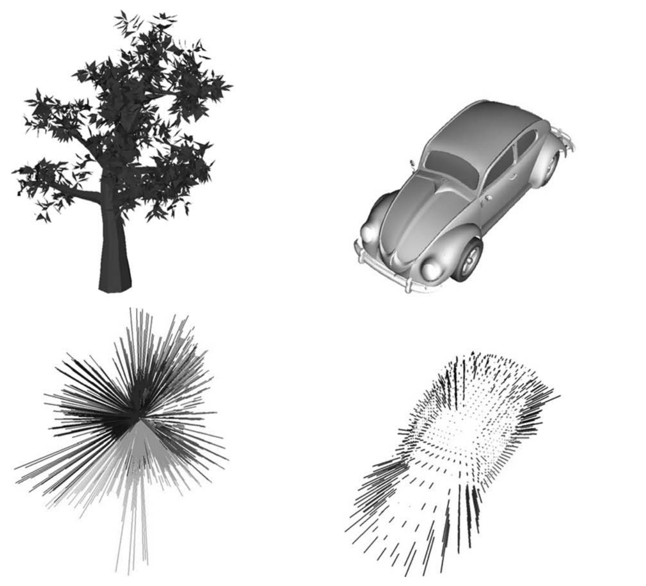
In the nonparallel case for both axes, the shortest distance and the corresponding perpendicular points P_1 and P_2 are determined. The test itself considers cone sections. One cone is intersected with a plane containing the first cone's center and the line through P_1 and P_2. This results in a well-known cone section. The first cone is reduced to a line, which passes the first cone's center and the point that you get by translating P_1 within the considered plane toward the cone section's focus. The intersection test is then reduced to a 2-D intersection test between a cone section and a line.

In the parallel case, the algorithm analyzes the projection of one cone onto the other cone's axis and vice versa. For each projection, two distance checks of points against their according radii are performed, which results in a total of four checks. The special case of parallel lines is separately handled for optimization, as in this case the whole test can be done by some interval checks, which are much faster.

Together with a quick rejection test (consider bounding cylinders instead of cones) and a quick acceptance test (consider spheres inside the cones) for the nonparallel case, the algorithm is reasonably fast.

Collision Detection Comparison: To show the efficiency of the new collision detection method, we have run a series of benchmarks on a Windows PC with a Pentium M 1.6-GHz processor. Realistically benchmarking collision detection algorithms is a difficult task. This is because there is a wealth of models with different characteristics and motions relative to each other. In [32], a benchmarking scheme for two models each contained in a unit box is proposed, where the second object performs a number of full-z rotations (in 1000 frames) at decreasing distances relative to the first object.

To compare this approach with well-known collision detection methods, we have included timings using an 18-sided discrete orientation polytope (18-DOP) [32] and oriented bounding boxes as in the publicly available library RAPID [33]. These approaches also report all triangle pairs in collision.



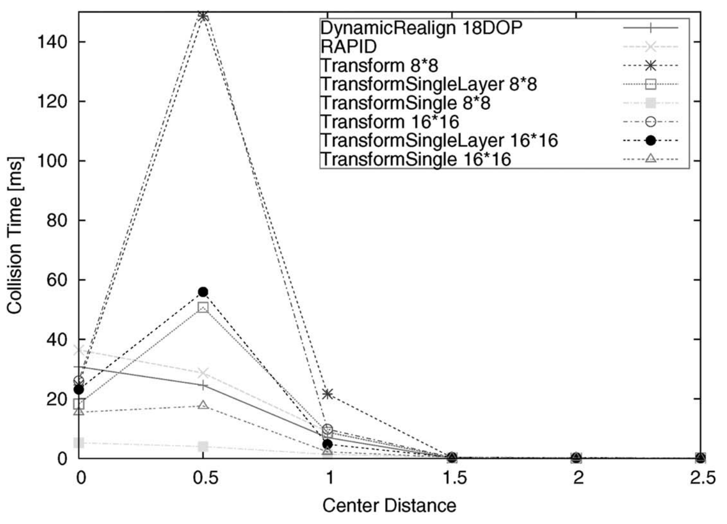


Fig. 7. Collision test for models Tree (4316 triangles) and Beetle (57 243 triangles). The line drawings show the axis of each capped cone of the discretized models. The benchmark "Transform" reports all triangle pairs in collision without elimination of duplicates; "TransformSingleLayer" reports a single triangle pair per layer of a colliding shell pair; and "TransformSingle" reports a single triangle pair in a colliding shell pair. For comparison, 18-DOP bounding volumes and oriented bounding volumes (public library RAPID) are shown.

The Fig. 7 contains the collision test models: a tree model colliding with a Beetle car model. This test demonstrates the algorithm's ability to handle all kinds of models. Absolutely unstructured polygon soups, as used for the leaves within the tree model, can be handled the same way like all other kinds of representations without any problems. Furthermore, the tree model is neither convex nor star shaped.

If reporting only a single triangle per spherical shell, the timings do not very much vary. In this mode of collision determination, the collision time is independent of the triangle count. It only depends on the sampling density and the volume between the inner and outer bounding shells.

V. Conclusion

We have presented a new collision detection algorithm, which is suitable for all model types, e.g., polygon soups, surfaces, and volumetric models. It is simple to implement, and its storage scheme consumes half the space compared to the full storage of the hierarchical spherical distance field.

The approach is scalable in the information it gives in collision determination. If it reports a single triangle per spherical shell, then the collision time only depends on the sampling density and the volume of spherical shells, which are used as bounding volumes, but not on the primitives' count of the model. Due to this fact, it is possible to tightly estimate the time bounds for the collision test. For bounding volume hierarchies, the worst-case time bounds are not tight in general, as the bounding volumes can arbitrarily overlap in space.

If we check all triangle pairs inside a spherical shell for intersection, then the approach works well in situations with few collisions, which are the most relevant in practice. In close-proximity situations, the algorithm degenerates to comparing triangle lists sorted according to the center distances of a triangle point.

Furthermore, we have shown that LOS calculation and collision detection can similarly be handled. Our wavelet-like storage scheme can be used with both arbitrary collision detection queries and LOS queries to create a more compact representation in memory.

References

[1] D. W. Fellner and N. Schenk, "MRT—A tool for simulations in 3D geometric domains," in *Proc. ESM*, Jun. 1997, pp. 185–188.
[2] A. Schmitz and M. Wenig, "The effect of the radio wave propagation model in mobile ad hoc networks," in *Proc. 9th ACM Int. Symp. Model., Anal. Simul. Wireless Mobile Syst. (MSWiM)*, 2006, pp. 61–67.
[3] M. Allegretti, M. Colaneri, R. Notarpietro, M. Gabella, and G. Perona, "Simulation in urban environment of a 3D ray tracing propagation model based on building database preprocessing," in *Proc. URSI Gen. Assem.*, 2005. [Online]. Available: www.ursi.org/Proceedings/ProcGA05/pdf/CP1.7(0958).pdf
[4] C. K. Yang, "Integration of volume visualization and compression: A survey," State Univ. New York, Stony Brook, NY, Tech. Rep. RPE-10, Aug. 2000.
[5] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross, "Optimized spatial hashing for collision detection of deformable objects," in *Proc. Vis., Model., Vis.*, Nov. 2003, pp. 47–54.
[6] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.
[7] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively sampled distance fields: A general representation of shape for computer graphics," in *Proc. SIGGRAPH*, 2000, pp. 249–254.
[8] S. Lefebvre and H. Hoppe, "Compressed random-access trees for spatially coherent data," in *Proc. EG Symp. Rendering*, Aug. 2007, pp. 339–350.
[9] M. D. Proctor and W. J. Gerber, "Line-of-sight attributes for a generalized application program interface," *JDMS*, vol. 1, no. 1, pp. 43–57, Apr. 2004.
[10] B. Salomon, N. Govindaraju, A. Sud, R. Gayle, M. Lin, D. Manocha, B. Butler, M. Bauer, A. Rodriguez, L. Eifert, A. Rubel, and M. Macedonia, "Accelerating line of sight computation using graphics processing units," in *Proc. 24th Army Sci. Conf.*, 2005. ADA433414. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA433414&Location=U2&doc=GetTRDoc.pdf
[11] D. Tuft, B. Salomon, S. Hanlon, and D. Manocha, "Fast line-of-sight computations in complex environments," Univ. North Carolina, Chapel Hill, NC, Tech. Rep. TR05-025, 2005.

[12] N. L. Max, "Horizon mapping: Shadows for bump-mapped surfaces," *Vis. Comput.*, vol. 4, no. 2, pp. 109–117, Mar. 1988.

[13] H. Rushmeier, L. Balmelli, and F. Bernardini, "Horizon map capture," *Comput. Graph. Forum*, vol. 20, no. 3, pp. 85–94, Sep. 2001.

[14] J. A. Stewart, "Fast horizon computation at all points of a terrain with visibility and shading applications," *IEEE Trans. Vis. Comput. Graph.*, vol. 4, no. 1, pp. 82–93, Jan. 1998.

[15] M. C. Lin and S. Gottschalk, "Collision detection between geometric models: A survey," in *Proc. 8th IMA Conf. Math. Surfaces*, 1998, pp. 37–56.

[16] M. C. Lin and D. Manocha, "Collision and proximity queries," in *Handbook of Discrete and Computational Geometry*. Boca Raton, FL: CRC Press, 2003.

[17] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino, "Collision detection for deformable objects," in *Proc. Eurographics*, 2004, pp. 119–140. State-of-the-Art Report.

[18] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six degrees-of-freedom haptic rendering using voxel sampling," in *Proc. SIGGRAPH*, 1999, pp. 401–408.

[19] A. Fuhrmann, G. Sobottka, and C. Groß, "Distance fields for rapid collision detection in physically based modeling," in *Proc. GraphiCon*, Sep. 2003, pp. 58–65.

[20] S. Fisher and M. C. Lin, "Deformed distance fields for simulation of non-penetrating flexible bodies," in *Proc. EG Workshop Comput. Animation Simul.*, 2001, pp. 99–111.

[21] D. Kidner, M. Dorey, and D. Smith, "What's the point? Interpolation and extrapolation with a regular grid DEM," in *Proc. GeoComputation*, 1999. [Online]. Available: http://www.geovista.psu.edu/sites/geocomp99/Gc99/082/gc_082.htm

[22] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding: An overview," *IEEE Trans. Consum. Electron.*, vol. 46, no. 4, pp. 1103–1127, Nov. 2000.

[23] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, "Wavelets for computer graphics: A primer, Part 1," *IEEE Comput. Graph. Appl.*, vol. 15, no. 3, pp. 76–84, May 1995.

[24] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, "Wavelets for computer graphics: A primer, Part 2," *IEEE Comput. Graph. Appl.*, vol. 15, no. 4, pp. 75–85, Jul. 1995.

[25] M. Akian, G. Cohen, S. Gaubert, R. Nikhoukhah, and J. P. Quadrat, "Linear systems in $(\max, +)$ algebra," in *Proc. 29th Conf. Decision Control*, Honolulu, HI, Dec. 1990, pp. 151–156.

[26] C. Fünfzig, T. Ullrich, D. W. Fellner, and E. N. Bachelder, "Empirical comparison of data structures for line-of-sight computation," in *Proc. IEEE Int. Symp. Intell. Signal Process. (WISP)*, 2007, vol. 1, pp. 291–296.

[27] C. Fünfzig, T. Ullrich, and D. W. Fellner, "Hierarchical spherical distance fields for collision detection," *IEEE Comput. Graph. Appl.*, vol. 26, no. 1, pp. 64–74, Jan./Feb. 2006.

[28] S. Valette and R. Prost, "Wavelet based multiresolution analysis of irregular surface meshes," *IEEE Trans. Vis. Comput. Graph.*, vol. 10, no. 2, pp. 113–122, Mar./Apr. 2004.

[29] H. Hoppe, "Efficient implementation of progressive meshes," *Comput. Graph.*, vol. 22, no. 1, pp. 27–36, Feb. 1998.

[30] K. Bühler, "Taylor models and affine arithmetics: Towards a more sophisticated use of reliable methods in computer graphics," in *Proc. Spring Conf. Comput. Graph.*, 2001, vol. 17, pp. 40–48.

[31] S. Krishnan, A. Pattekar, and M. C. Lin, "Spherical shell: A higher order bounding volume for fast proximity queries," in *Proc. 3rd Workshop Algorithmic Found. Robot.*, 1998, vol. 3, pp. 177–190.

[32] G. Zachmann, "Rapid collision detection by dynamically aligned DOP-trees," in *Proc. Virtual Reality Annu. Int. Symp.*, 1998, pp. 90–97.

[33] S. Gottschalk, M. C. Lin, and D. Manocha, "OBB-tree: A hierarchical structure for rapid interference detection," in *Proc. SIGGRAPH*, 1996, pp. 171–180.

Christoph Fünfzig received the M.Sc. degree from the University Bonn, Bonn, Germany, and the Ph.D. degree in computer science from Braunschweig University of Technology, Braunschweig, Germany.

He is currently with the CAGD Group, Univ. Valenciennes et du Hainaut-Cambrésis, FR CNRS 2956, Valenciennes, France. His research interests include practical computational geometry, animation and simulation in computer graphics, virtual/augmented reality, and scientific visualization.

Torsten Ullrich received the M.Sc. degree in mathematics from Karlsruhe Institute of Technology, Karlsruhe, Germany. He is currently working toward the Ph.D. degree in computer science with Graz University of Technology, Graz, Austria.

His research has been concerned with computer-aided geometric design topics, including modeling and reconstruction.

Dieter W. Fellner received the M.Sc. and Ph.D. degrees from Graz University of Technology, Graz, Austria.

He is the Director of the Fraunhofer Institute of Computer Graphics (IGD), Darmstadt, Germany, and a Professor of computer science with Darmstadt University of Technology (TU Darmstadt), Darmstadt, with a joint affiliation with Graz University of Technology. His research interests include computer graphics, modeling, immersive systems, and graphics in digital libraries. He has held academic positions with universities in Graz, Austria; Denver, CO; St. John's, NF, Canada; Bonn, Germany; and Braunschweig, Germany.

Dr. Fellner is a member of the Association for Computing Machinery, Eurographics, and Gesellschaft für Informatik (GI). He is on the Editorial Board of several international journals, IEEE COMPUTER GRAPHICS AND APPLICATIONS being one of them.

Edward N. Bachelder received the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge.

Prior to receiving his Ph.D. degree, he was a Naval Aviator flying the SH-60B. He is currently a Principal Research engineer with Systems Technology Inc., Hawthorne, CA. His areas of research include augmented reality, optimized control guidance for helicopter autorotation training and operation, real-time path optimization, system identification (extremely low to very high frequency regimes) using sparse excitation, 3-D helicopter cueing for precision hover, and nap-of-earth flight.