Fast Computation of Projections from Triangulated Surfaces

S. Sawall[1,2], M. Baer[1,2], M. Brehm[1,2], M. Knaup[1], and M. Kachelrieß[1,2]

[1]Medical Physics in Radiology, German Cancer Research Center (DKFZ), Heidelberg, Germany

[2]Institute of Medical Physics, Friedrich-Alexander-University (FAU), Erlangen, Germany

Purpose

The forward projection is an operation widely used in computed tomography (CT) and other imaging modalities. In most cases voxels are used as basis functions within the employed algorithms. The design of complex anthropomorphic phantoms using voxels is very time consuming. Among others, computer aided design (CAD) software provides triangulated objects with almost arbitrary complexity that might be used as phantoms. The forward projection of such a triangulated object requires the calculation of all intersection points of a desired ray in CT or MR or a line of response in SPECT or PET with all triangles of the object (see figure 1). As several billion rays (at least for CT) have to be computed to obtain a complete dataset this method is only feasible for small objects. We propose a fast method for the calculation of intersection lengths of highly complex objects containing millions of triangles. This method employs a spatial subdivision scheme to speed up the computations to provide projections of triangulated objects for reconstruction and artifact correction methods.

Materials and Methods

We herein assume that the triangulated surfaces are provided and will not consider their generation. See reference [1] for additional details. Thus, the intersection length of each ray with these surfaces needs to be calculated and weighted with an attenuation coefficient to obtain projection images. As it is not computationally reasonable to calculate the intersection points of all triangles with all simulated rays, a spatial subdivision structure is used to speed up these computations. In particular an octree is used in the following. An octree in three-dimensional space is a tree structure with each node in general containing eight child nodes. If we consider the root node to be the bounding box of our triangulated object these child nodes correspond to equally sized spatial subdivisions, i.e. rectangular boxes, of this bounding box. These nodes will be referred to as internal nodes in the following. This subdivision process continues by subdividing each child node into eight boxes again. If the space enclosed by such a child does not contain any triangles, it will no further be used to spawn new child nodes. I.e., its parent node rejects this child. This process is illustrated in figure 2. This subdivision in our case stops as soon as no new boxes can be generated that contain at least 16 triangles. The triangles contained in a subbox are stored in an external child node. When the intersections of a ray with an object shall be computed, the octree is traversed. First it is ensured that the ray intersects with the root node.

Applications

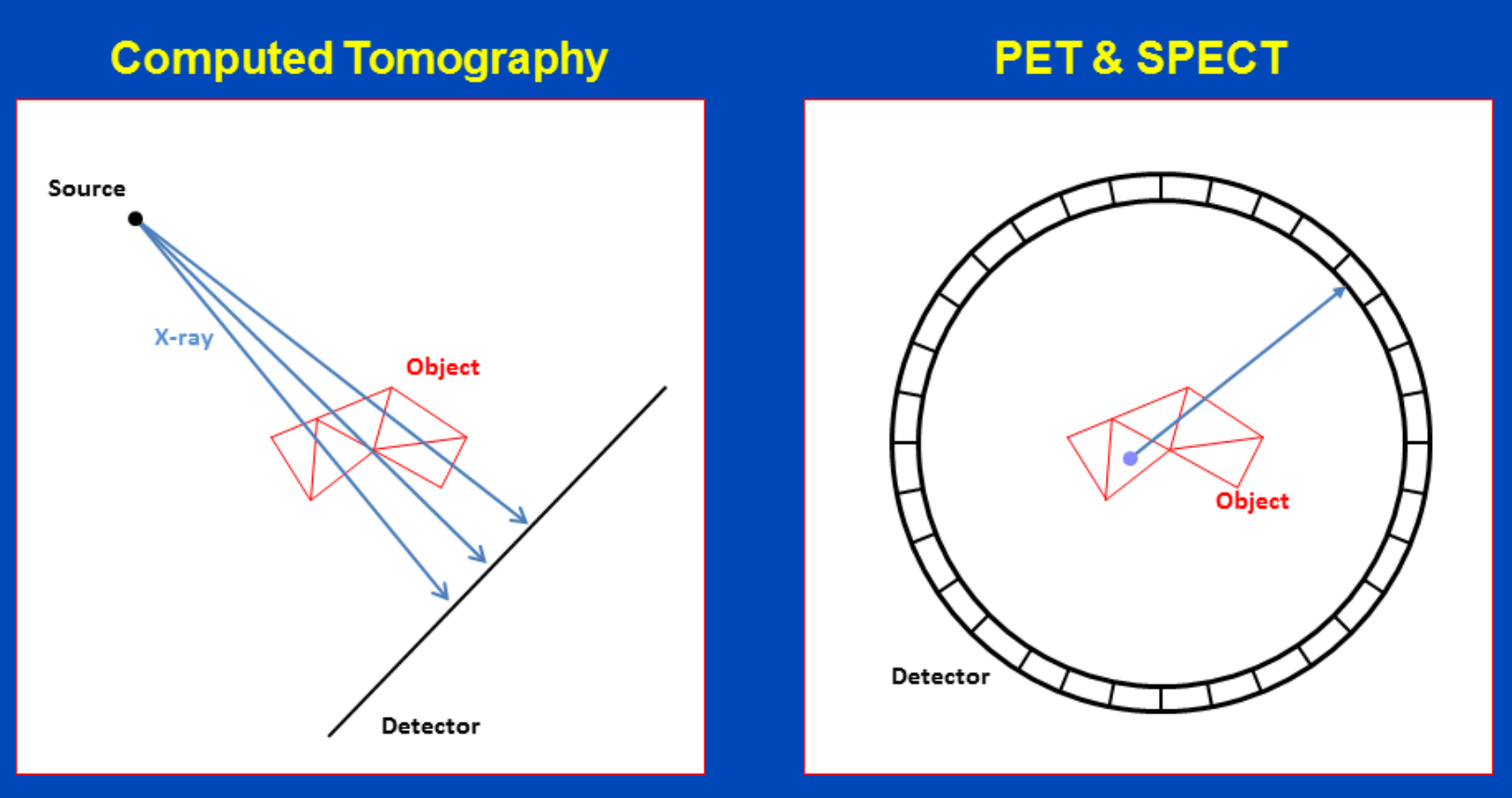

Figure 1: Possible applications of the proposed technique are artifact correction methods or advanced simulations in Computed Tomography, Positron Emission Tomography and other medical imaging modalities.

Octree

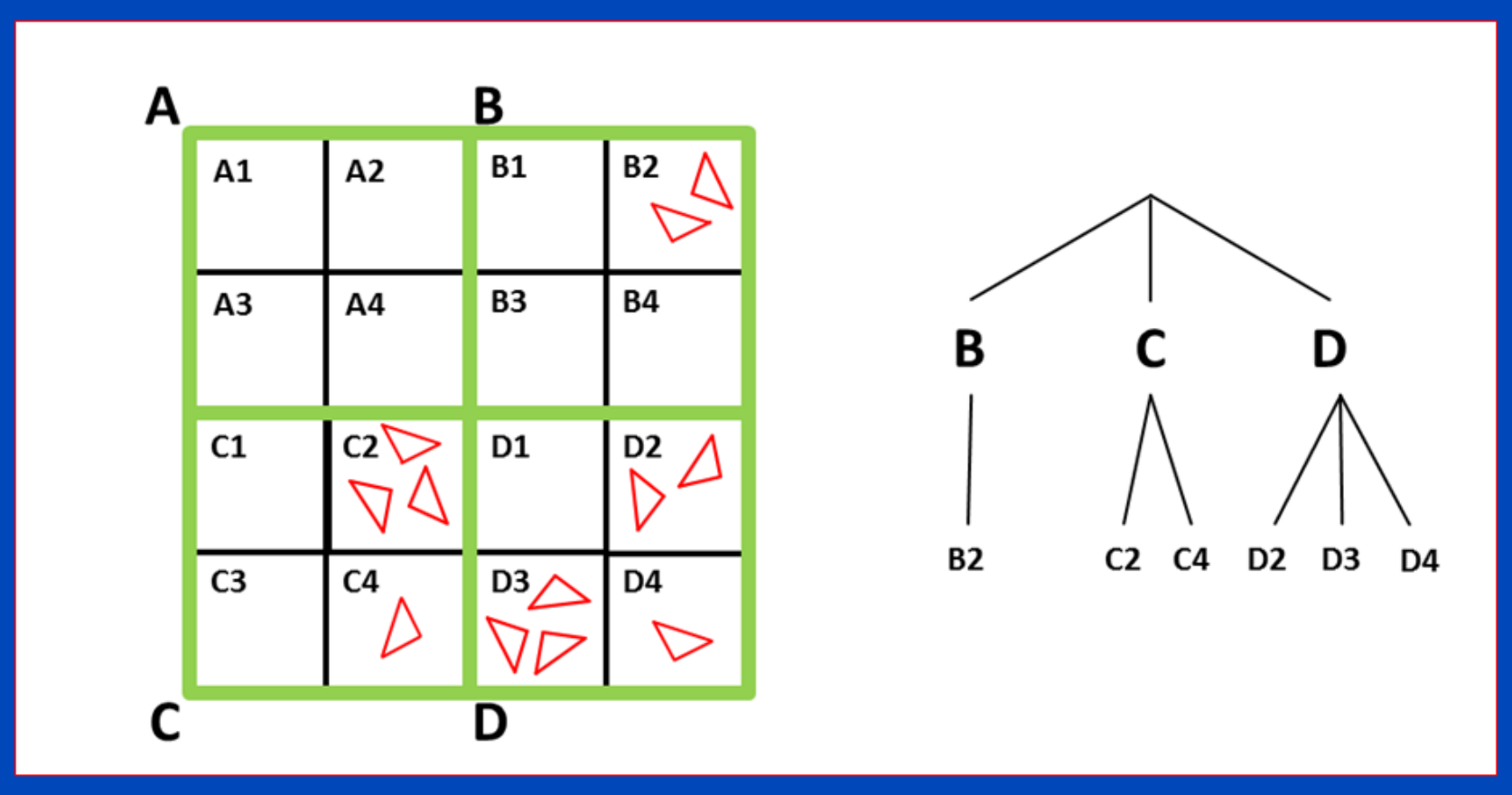

Figure 2: Left part of the figure exemplarily shows the created subboxes corresponding to some triangles marked in red. The right hand-side illustrates the tree-representation of this spatial subdivision. Note that empty cells are not considered in the data structure.

Octree Traversal

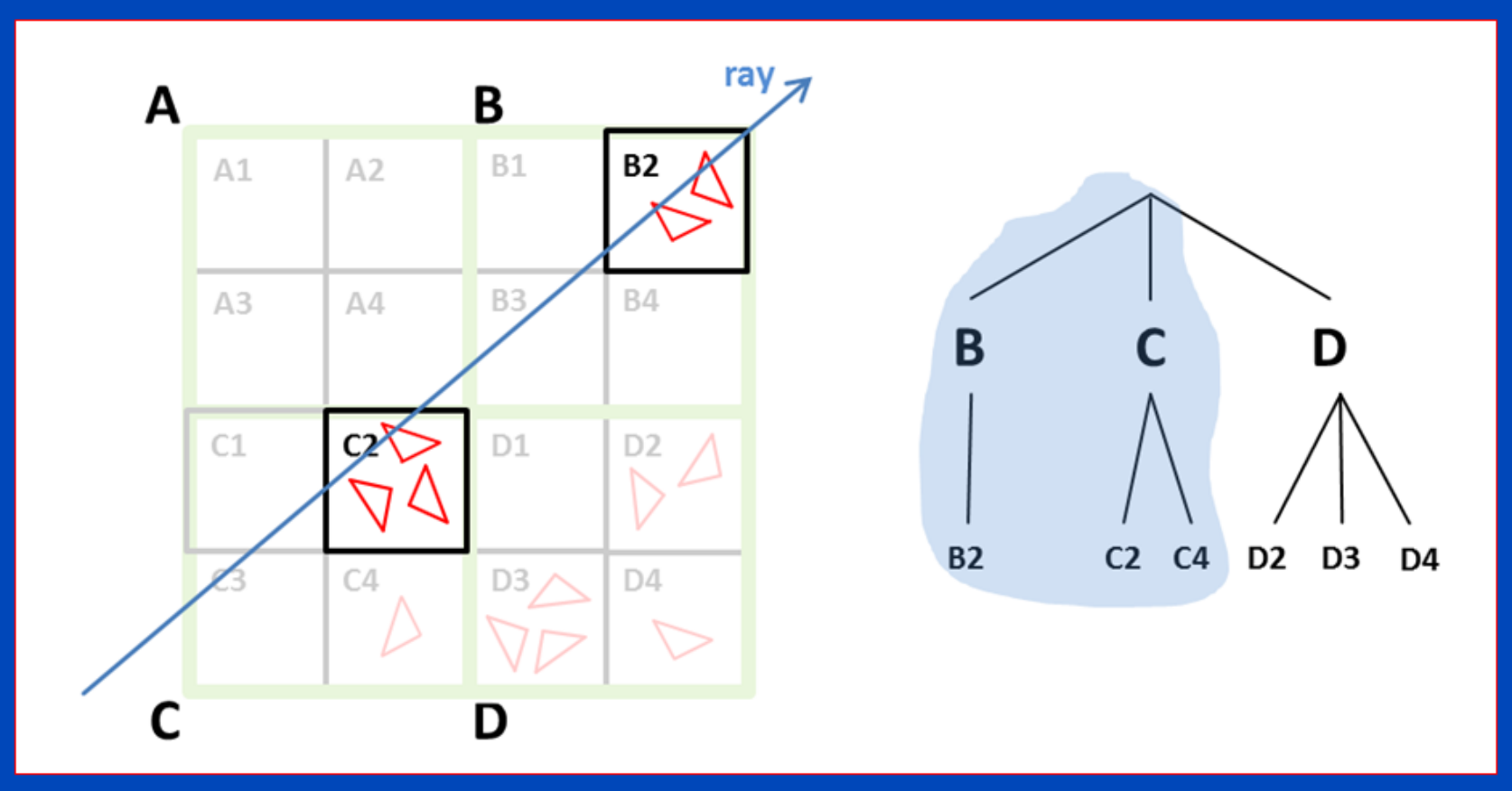

Figure 3: If the intersections of a ray with the object shall be computed, the octree is traversed. Due to the spatial subdivision only the triangles in the highlighted bounding boxes need to be considered for an intersection allowing for a high performance.

Simulation Parameters

- Simulated scan:
  - Source-isocenter-distance: 572 mm
  - Isocenter-detector-distancer: 375 mm
  - Flat detector with 1000×1000 pixels each of 388 µm
  - 720 projections over 360°
  - $7.2 \cdot 10^8$ ray-object tests per dataset
  - Brute force: typical object with 10^6 triangles results in $7.2 \cdot 10^{14}$ ray-triangle intersection tests
- Simulated objects:
  - Object 1: Single sphere with a diameter of 35 mm
  - Object 2: Grid (3×3×3) of spheres with a diameter of 15 mm each
  - The vertices of each sphere are perturbed by Gaussian noise with a standard deviation of 1.0 mm
- Test system:
  - 1 workstation with two Intel Xeon hexacore processor @ 3.46 GHz
  - 96 GB of RAM
  - Hyperthreading was disabled

Figure 4: Scan and simulation parameters used to assess the performance of the proposed method.

Test Objects: Intersection Lengths

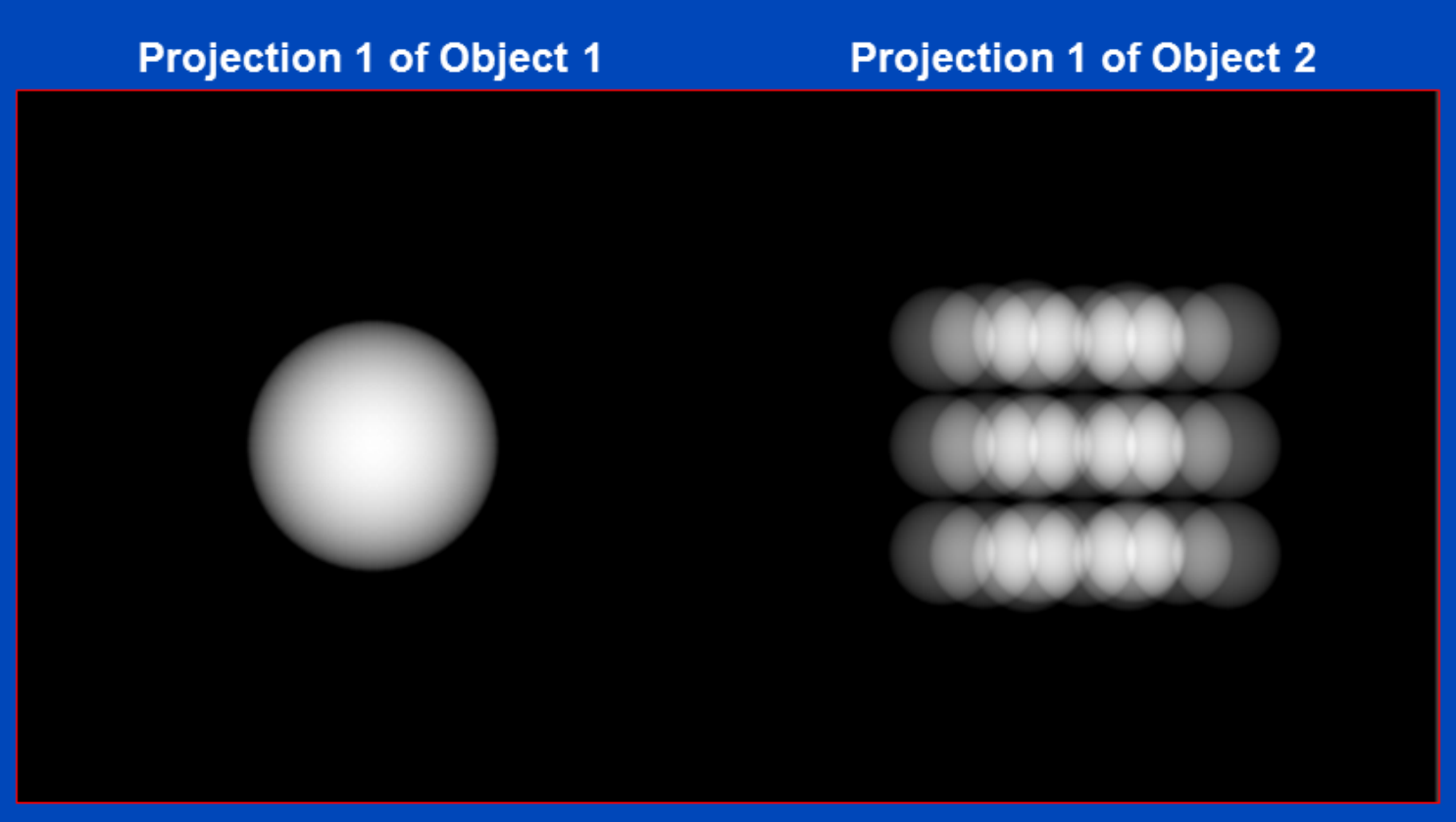

Figure 5: Intersection lengths obtained from the two test objects using $25 \cdot 10^6$ triangles per object.

Performance

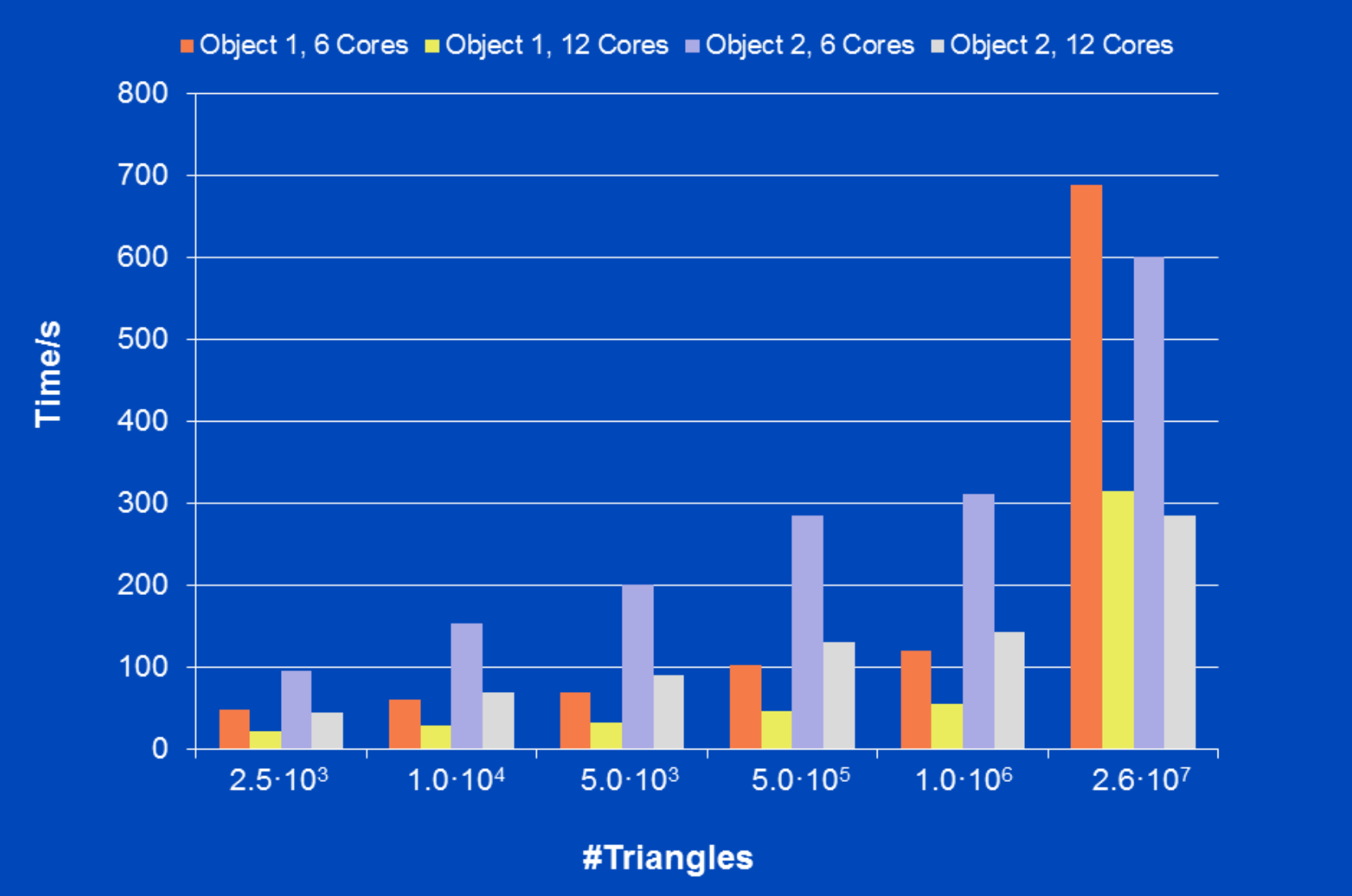

Figure 6: Performance as function of the used CPU cores and the number of triangles.

Performance: Object 1

Cores \ Triangles	$2.5 \cdot 10^3$	$1.0 \cdot 10^4$	$5.0 \cdot 10^4$	$5.0 \cdot 10^5$	$1.0 \cdot 10^6$	$2.6 \cdot 10^7$
1	246 s	309 s	355 s	529 s	610 s	3474 s
4	63 s	80 s	91 s	136 s	158 s	910 s
6	48 s	61 s	69 s	103 s	120 s	689 s
8	32 s	41 s	47 s	70 s	82 s	468 s
12	22 s	28 s	32 s	47 s	55 s	315 s

Figure 7: Runtime of the proposed method in seconds using one to 12 CPU cores. The number of triangles in object one , a procedurally generated sphere, was varied between 2500 and $26 \cdot 10^6$.

Performance: Object 2

Cores \ Triangles	$2.5 \cdot 10^3$	$1.0 \cdot 10^4$	$5.0 \cdot 10^4$	$5.0 \cdot 10^5$	$1.0 \cdot 10^6$	$2.6 \cdot 10^7$
1	500 s	790 s	1040 s	1481 s	1610 s	3088 s
4	125 s	205 s	265 s	375 s	412 s	776 s
6	95 s	154 s	200 s	285 s	312 s	601 s
8	65 s	103 s	135 s	194 s	212 s	426 s
12	44 s	70 s	91 s	131 s	143 s	285 s

Figure 8: Runtime of the proposed method in seconds using one to 12 CPU cores. The number of triangles in object two, a procedurally generated sphere grid, was varied between 2500 and $26 \cdot 10^6$.

Medical Example

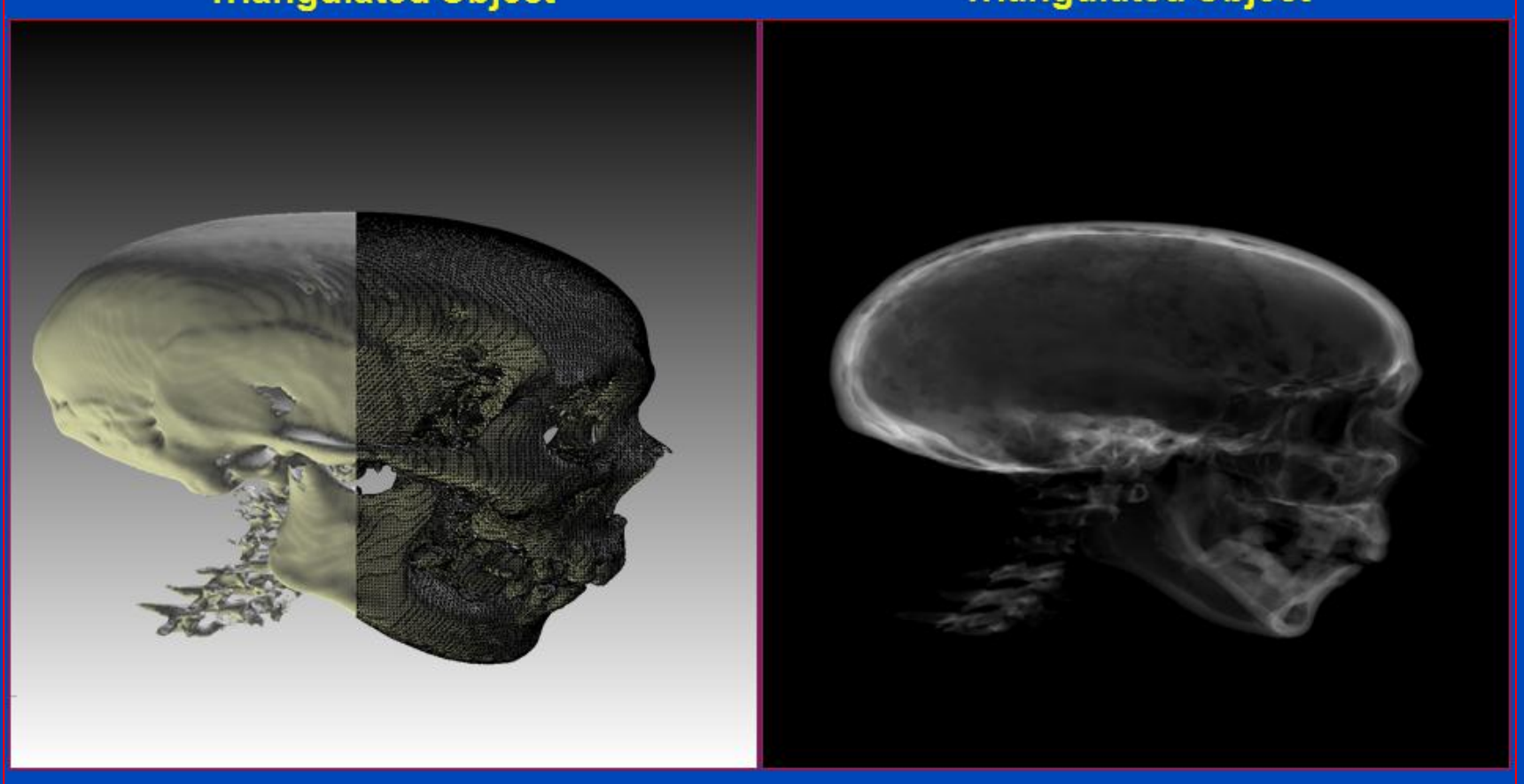

Figure 9: A triangulated skull (700k triangles) obtained from a CT scan and the projection image computed therefrom. A complete dataset (720 projections each 1000² pixels) required 42 s.

Comparison to Other Approaches

- Reference [2] provides a GPU-based method for the computation of intersection lengths from triangulated surfaces using the following simulation parameters:
  - Flat detector with 640×480 pixels
  - Object with $4.6 \cdot 10^5$ triangles
  - Time per projection is 173 ms
- If we assume that this method scales linear with the number of triangles and detector pixels, the simulation of a projection with 1000×1000 pixels from an object with $5 \cdot 10^5$ triangles would roughly require 612 ms.
- This is significantly slower than the reported 65 ms (47 s for 720 projections using 12 CPU cores) in figure 7.

Figure 10: Comparison of the proposed method to other algorithms presented in the literature.

If this is not the case the procedure is terminated. If the ray intersects the root node all internal child nodes are recursively checked for an intersection (see figure 3). If any of these nodes contains an external child node the intersection points with the triangles enclosed therein are computed. Using this spatial subdivision scheme ensures that the number of triangles that have to be tested for an intersection with a ray is highly reduced and thus performance is increased. To illustrate the capabilities of the proposed method two test objects have been designed. Both test objects are procedurally generated and allow to dynamically increase the number of triangles.

Results

Figure 5 shows intersection lengths computed by the proposed method for the two test objects. Figures 6, 7 and 8 present the runtime of the proposed method in seconds for both test cases, different triangle counts per test case and different numbers of CPU cores used to parallelize the computations. Note that the creation of the octree is not included in these measurements. It roughly corresponds to about 5% of the reported runtimes. In the worst case of $26 \cdot 10^6$ triangles per test case a complete dataset can be obtained in less than 3500 seconds in all cases. Note that typical datasets only contain one to two million triangles, resulting in a complete dataset within less than a minute. Although both test cases provide the same triangle counts the total runtime differs significantly. This is caused by the fact that the octree contains more nodes if multiple, spatially disjunct objects are used and thus the time for tree traversal is increased. A real world example is presented in figure 9. The method implemented on the CPU is further faster than recently published work employing the GPU (see figure 10).

Conclusion

We proposed a method for the computation of intersection lengths from triangulated surfaces. It was possible to demonstrate that intersection lengths of objects containing several million triangles can be calculated in a few seconds using modern CPUs. The proposed algorithm scales almost linear with the number of processor cores and thus provides highest performance, e.g. for the computation of projections from anthropomorphic phantoms. This allows for the usage of triangulated objects in simulations, image reconstruction and artifact correction in Computed Tomography and other imaging modalities.

References

[1] W. E. Lorensen and H. E. Cline, “Marching Cubes: A high resolution 3D surface construction algorithm,”SIGGRAPH Comput. Graph., vol. 21, no. 4, pp. 163–169, 1987.

[2] A. Maier, H. G. Hofmann, C. Schwemmer, J. Hornegger, A. Keil, and R. Fahrig, “Fast simulation of x-ray projections of spline-based surfaces using an append buffer,” Physics in Medicine and Biology, vol. 57, no. 19, pp. 6193–6210, 2012.

Send correspondence request to:
Dr. Stefan Sawall
stefan.sawall@dkfz.de
German Cancer Research Center (DKFZ)
Im Neuenheimer Feld 280
69120 Heidelberg, Germany

dkfz.

Fully3D 2013