Appendix A: Supporting the Quadric Error Metric

Our simplification method is (half-)edge collapse using the (extended) quadric error metric (QEM) [GZ05], where for an edge (a,b) we either collapse from a to b or b to a, whichever is allowed and has the smaller error. When simplifying the current sub-volume, we put all candidates into a priority queue so that we simplify from the smallest to largest errors (in contrast to the multiple-choice randomized edge collapse of [VCL*07]). If the current edge e cannot be collapsed, we put it aside and later put it back to the priority queue when its neighborhood condition changes and e can be reactivated. To support QEM, for each vertex v we keep a 15-scalar error matrix using the information of all tetrahedra incident on v, and an additional 10-scalar error matrix if v is on the mesh boundary. When a vertex a is collapsed to v, their corresponding matrices are added component-wise [GZ05]. Since we simplify one sub-volume at a time, special care is needed to handle the *shared* vertices between neighboring sub-volumes.

Initially we compute the QEM for leaf sub-volumes one by one. For each shared vertex, we additionally sum up the matrices from neighboring sub-volumes, and save (global vertex ID, quadric error matrix/matrices) of the shared vertices in a separate file F.

At the beginning of the simplification, we load and keep F in main memory throughout the entire process; F is small and can entirely fit (at most 551MB in all our experiments). When we need to simplify the sub-volume boundary involving a shared vertex, F is consulted and updated accordingly (using the global vertex ID as the key). Since we only remove vertices, the size of F can only shrink. In this way we can easily support the quadric error metric.

Appendix B: Proof of the Lemma

Proof: Recall that this LOD cut on the tree M must be the root cut of some construction stage (see Fig. 3 and Section 3.2.2). Letting ω be the node in this cut with the largest ε_l value, we see that ω is the new node created at this construction stage. We have made this cut crack-free by making the boundaries consistent at this stage. Therefore, all we need to do is to update the boundaries to the point that they return to the same boundary status of this stage, namely, to apply all the marked sequences $S_1, S_2, \cdots$ up to and including the marked sequence propagated by ω. Note that ω is the node in the cut with the largest ε_l value, i.e., $\varepsilon_l(\omega) \leq \varepsilon$. Therefore we apply all the marked sequences $S_1, S_2, \cdots$ until the first S_i whose mark is $\varepsilon_l(\omega_i) > \varepsilon$. This means that ω_i is *not* in the cut and is created *after* ω, and thus the boundary updates should stop there. □

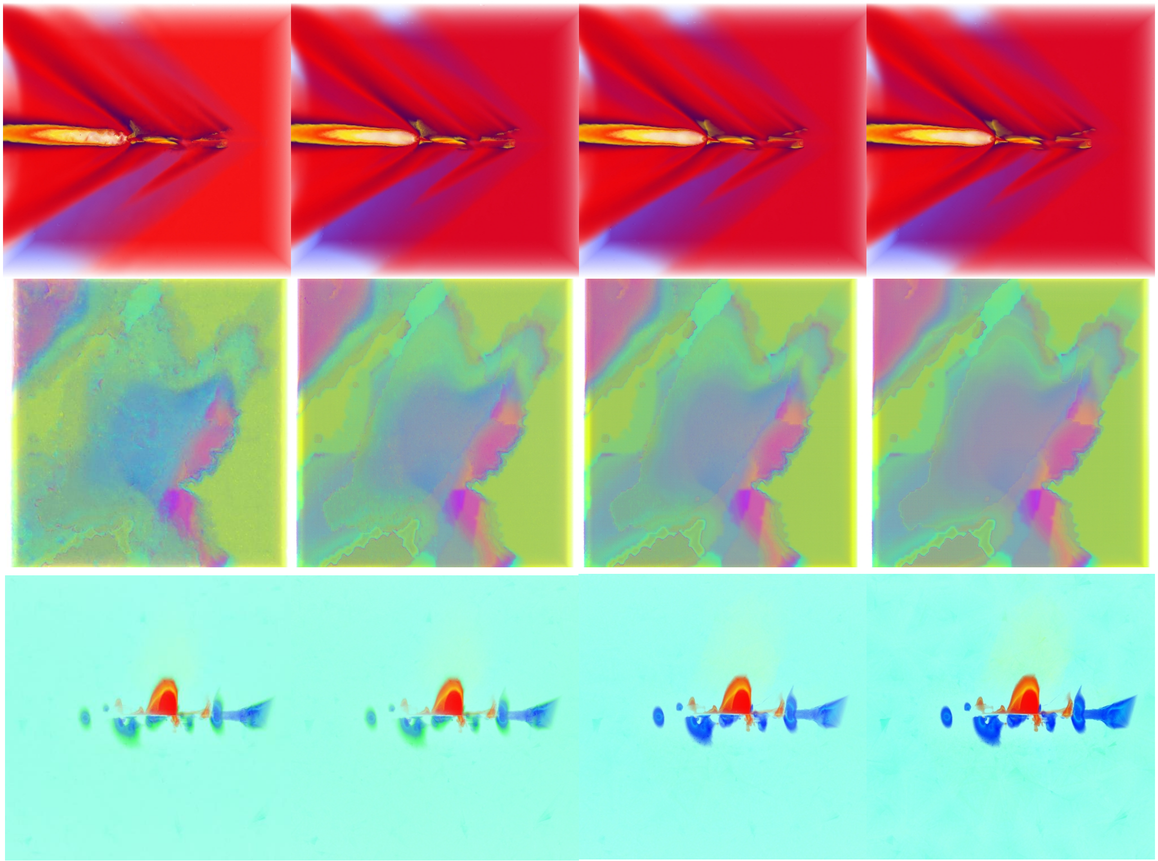

Figure 6: *Representative images of our out-of-core* uniform *LOD volume rendering, corresponding to the queries of Fig. 4 (with increasing LOD resolutions) from left to right. The rightmost column has 100% resolution (i.e., $\varepsilon = 0$). Top row: Fighter; middle row: SF1-x; bottom row: F16-x.*