Mesh-based iterative reconstruction for computed tomography

Junho Lee, Yoonsang Hong, Seungjun Yoo, Seokwon Oh, and Ho Kyung Kim *
Computational X-ray Imaging Laboratory, School of Mechanical Engineering, Pusan National University, Busandaehak-ro 63beon-gil, Geumjeong-gu, Busan 46241, Republic of Korea

Keywords:** ***Mesh-based CT reconstruction, Iterative reconstruction, ray tracing

1. Introduction

Industrial X-ray CT is used not only for non-destructive inspection but also for shape acquisition and generating geometric data for physics-based simulations [1]. However, voxel-based CT representations can become extremely large, increasing the burden of rendering, post-processing, and downstream applications.

Moreover, voxel faces and edges often do not align with true material interfaces; voxels near boundaries may contain multiple materials, causing partial volume effects (PVE) and degrading interface localization accuracy [2]. While using a finer voxel grid can mitigate PVE, it typically increases memory usage [2]. In industrial workflows, surfaces are frequently extracted from CT volumes as meshes (e.g., STL) [1], and a series of conversions from CT reconstruction to surface extraction can reduce surface accuracy—thus minimizing conversion steps is desirable [1].

Motivated by these issues, mesh-based reconstruction has been explored, where attenuation values are estimated directly on unstructured meshes such as triangles/tetrahedra [1-3]. Meshes can represent interfaces more precisely because vertices are not constrained to grid coordinates [2], and adaptive meshing can reduce the number of unknowns and memory footprint across iterations (coarser in homogeneous regions and finer near boundaries) [3]. Mesh-based representations also support workflows that reduce reliance on voxel-to-mesh conversion (e.g., marching cubes) [1, 2].

A key practical requirement is repeated evaluation of projection (ray-element intersection lengths and accumulated line integrals) on unstructured meshes. This research presents an implementation of tetrahedral-mesh iterative reconstruction using NVIDIA OptiX API, focusing on an OptiX-based projector/backprojector. OptiX is a GPU ray-tracing framework that builds BVH acceleration structures and allows user-derafined programs (e.g., *any-hit*, *closest-hit*, *intersection*) to inject custom intersection handling and accumulation into the ray traversal pipeline. Using these capabilities, we integrate ray-element accumulated line integration on unstructured tetrahedral meshes into a ray-tracing workflow.

2. Methods and Materials

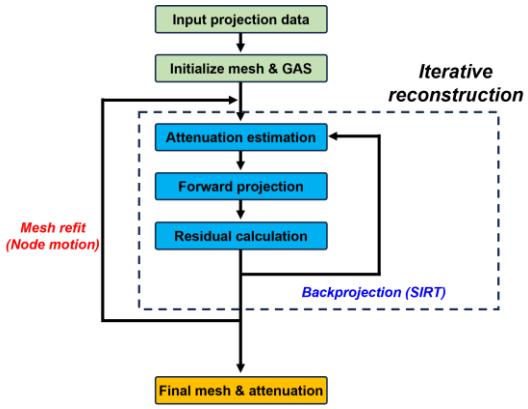

Fig. 1. Illustration of Mesh-based iterative reconstruction pipeline

2.1 OptiX ray tracing

We model the object as a tetrahedral mesh where the attenuation coefficient is assumed constant per tetrahedron. For each detector pixel (ray) i, the predicted log-projection is computed as a line integral discretized by tetrahedron intersection lengths

$$\tau_i^{\text{pred}} = \sum_t \mu_t l_{i,t}, \qquad L_i = \sum_t l_{i,t}, \tag{1}$$

Where $l_{i,t}$ is the ray intersection length inside tetrahedron t and L_i the total traversed length used for row normalization.

To evaluate $l_{i,t}$ on an unstructured tetrahedral mesh, we implement the projector/backprojector using NVIDIA OptiX. Each tetrahedron is represented as a custom primitive with an associated axis-aligned bounding box (AABB), enabling OptiX to build a BVH acceleration structure for fast candidate traversal. In the custom intersection program, the ray-tetrahedron interval $[t_{\text{enter}}, t_{\text{exit}}]$ is computed by evaluating ray-plane intersections with the four boundary planes [4]. The values $(t_{\text{enter}}, t_{\text{exit}})$ are passed to the *any-hit* program as attributes.

In the *any-hit* program, the intersection length is computed as

$$l_{i,t} = \max(0, t_{\text{exit}} - t_{\text{enter}}), \tag{2}$$

By allowing rays to fully traverse the volumetric mesh, the system computes the line integral by accumulating contributions from all intersected tetrahedra.

The ray generation program launches one ray per detector pixel using the per-view geometry $(\mathbf{s}, \mathbf{c}, \mathbf{u}, \mathbf{v})$, where $\mathbf{s}$ is source position, $\mathbf{c}$ is the detector center, and $\mathbf{u}$, $\mathbf{v}$ are detector basis vectors. The input projections are provided as log-transformed measurements $\tau^{\text{meas}} = -\log(I/I_0)$, consistent with the implemented residual computation.

2.2 SIRT

We employ the simultaneous iterative reconstruction technique (SIRT) on the tetrahedral unknowns $\{\mu_t\}$. Let $\mathbf{b}$ denote the measured log-projections and $\mathbf{A\mu}$ the predicted log-projections. In the mesh-based discretization, the system matrix entries are defined by ray-tetrahedron intersection lengths

$$(\mathbf{A\mu})_i = \sum_t A_{i,t}\mu_t, \qquad A_{i,t} = l_{i,t}, \tag{3}$$

At each iteration, OptiX-based ray tracing is used to compute the predicted log-projection τ_i^{pred} and the total traversed length L_i required for row normalization. The per-ray residual is then defined as

$$r_i = b_i - \tau_i^{\text{pred}}, \tag{4}$$

Using a row-normalized residual can mitigate ray-length-dependent scaling

$$r_i^{\text{norm}} = \frac{r_i}{L_i+\epsilon}, \tag{5}$$

Next, we trace the same rays again and perform tetrahedron-wise backprojection by accumulating length-weighted contributions within any-*hit* stage. Specifically, for each tetrahedron t, a correction term g_t and a normalization weight w_t are calculated as follows

$$g_t \mathrel{+}= l_{i,t} r_i^{\text{norm}}, \qquad w_t \mathrel{+}= l_{i,t}, \tag{6}$$

Finally, each tetrahedral attenuation coefficient is updated with relaxation factor α and non-negativity is enforced

$$\mu_t^{k+1} = \max\left(0, \mu_t^k + \alpha \frac{g_t}{w_t+\epsilon}\right), \tag{7}$$

2.3 Node motion

An optional node-motion step is included, where vertex positions $\mathbf{p}$ are updated while preserving tetrahedral topology. The motion is formulated by converting element-wise estimates into vertex-associated scalar values and applying constrained vertex displacements to better represent material interfaces.

For each vertex p, a vertex-averaged attenuation $\bar{\mu}(p)$ and a vertex normalization weight $w(p)$ are computed from the incident tetrahedra $\mathrm{T}(p)$

$$\bar{\mu}(p) = \frac{1}{|\mathrm{T}(p)|}\sum_{t\in \mathrm{T}(p)} \mu_t, \qquad w(p) = \sum_{t\in \mathrm{T}(p)} w_t, \tag{8}$$

Where w_t denotes the per-tetrahedral normalization weight accumulated during SIRT backprojection (Section 2.2).

For interior vertices, $\nabla\bar{\mu}$ is approximated from a local neighborhood. An edge-attraction displacement is defined using a neighborhood average μ_{avg}

$$\bar{\mu}_{\text{avg}} = \frac{1}{|N(p)|}\sum_{q\in N(p)} \bar{\mu}(q), \tag{9}$$

$$s = \text{clip}\left(\frac{\bar{\mu}(p)-\bar{\mu}_{\text{avg}}}{\|\nabla\bar{\mu}(p)\|}, -s_{\max}, s_{\max}\right), \tag{10}$$

$$\Delta\mathbf{p}_{\text{edge}} = -s\frac{\nabla\bar{\mu}(p)}{\|\nabla\bar{\mu}(p)\|}, \tag{11}$$

Where, $N(p)$ denotes the node neighborhood, and $\text{clip}(\cdot)$ limits the per-update displacement by $s_{\max}$.

To regularize mesh distortion, an edge-preserving weighted Laplacian smoothing term is added

$$\Delta\mathbf{p}_{\text{smooth}} = \lambda w(g)(\|\nabla\bar{\mu}(p)\|)\left(\mathbf{p}_{\text{avg}} - \mathbf{p}\right), \tag{12}$$

$$w(g) = \frac{\rho_0}{g+\rho_0}, \tag{13}$$

Where $\mathbf{p}_{\text{avg}}$ is the average of neighboring vertex positions. The final displacement $\Delta\mathbf{p} = \Delta\mathbf{p}_{\text{edge}} + \Delta\mathbf{p}_{\text{smooth}}$ is clamped to $\|\Delta\mathbf{p}\| \le s_{\max}$ before being applied to the vertex buffer.

After node motion, tetrahedron AABBs are recomputed and the OptiX geometry acceleration structure is refit, enabling subsequent ray tracing to reflect the deformed geometry.

3. Preliminary Results

Fig. 2. (a) Gear phantom (CAD model). (b) Simulated transmission projection generated using OptiX-based forward projector with a 450 kV spectrum and a 2 mm Sn filter

Fig. 2(b) shows a simulated transmission projection generated from the gear phantom in Fig. 2(a) using the OptiX-based forward projector. A polyenergetic spectrum corresponding to 450 kV with a 2 mm Sn filter was used.

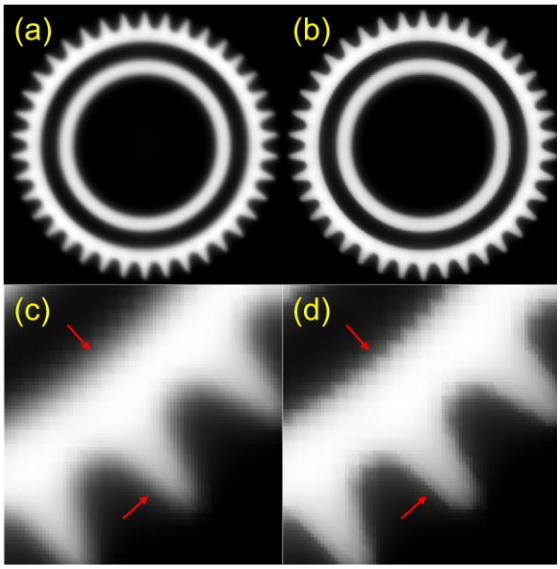

Fig. 3. Reconstruction comparison on a tetrahedral mesh obtained by subdividing a regular voxel grid into six uniform tetrahedra per voxel. (a, c) Reconstruction without node motion. (b, d) Reconstruction with node motion.

Fig. 3 compares mesh-based iterative reconstructions performed on a tetrahedral mesh obtained by subdividing a regular voxel grid into six uniform tetrahedra per voxel. The reconstruction without node motion (Fig. 3(a, c)) exhibits visibly blurred material boundaries. When node motion is enabled (Fig. 3(b, d)), sharper boundary transitions and improved edge localization are observed in both the full view and the magnified region. Since the number of tetrahedra is unchanged, this improvement indicates that geometry adaptation (vertex displacement) provides better boundary representation under the same memory budget.

The observed improvement is consistent with the node-motion design: vertex displacements are driven by local gradients of the vertex-averaged attenuation field, so updates are preferentially activated near high-gradient interface regions and tend to move vertices toward a locally consistent interface value. While rigorous quantitative validation remains to be conducted, the visual results suggest that the vertex positions were effectively adjusted toward interface-dominant regions, leading to reduced boundary smearing.

4. Conclusion and Future Study

Future work will extend the current approach by introducing mesh refinement/splitting (adaptive subdivision) in addition to vertex motion. Adaptive refinement is expected to (i) allocate degrees of freedom preferentially near interfaces and small defects (e.g., cracks) and (ii) reduce redundant unknowns in homogeneous regions, enabling higher fidelity reconstruction under constrained memory. Further studies will also include quantitative evaluation and a broader range of acquisition settings.

ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2024-00340520). J. Lee and Y. Hong were supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Climate, Energy and Environment (MCEE) of the Republic of Korea (No. RS-2024-00398425).

REFERENCES

[1] Y. Nagai, Y. Ohtake, and H. Suzuki, CT Reconstruction on Unstructured Mesh for Multi-material Object, in *Proc. 19th World Conference on Non-Destructive Testing (WCNDT)*, 2016.

[2] J. Merckx, A. J. den Dekker, J. Sijbers, and J. De Beenhouwer, DAMMER: Direct Adaptive Multi-Resolution MEsh Reconstruction From X-Ray Measurements, *IEEE Transactions on Computational Imaging*, vol. 11, pp. 926–941, 2025.

[3] F. Buyens, M. A. Quinto, and D. Houzet, Adaptive Mesh Reconstruction in X-Ray Tomography, in *Proc. MICCAI Workshop on Mesh Processing in Medical Image Analysis (MeshMed)*, 2013.

[4] M. Cyrus and J. Beck, Generalized two- and three-dimensional clipping, *Computers & Graphics*, vol. 3, no. 1, pp. 23–28, 1978.