High-Quality Analytic Isosurface Rendering for Web-Based Cardiac Imaging

Vasileios Lazaros Charalampidis*, M. Louis Handoko†, Eduard Ródenas-Alesinao‡,
Folkert W. Asselbergs§, Konstantinos Votis*, Paschalis Bizopoulos*, Andreas Triantafyllidis*
*Information Technologies Institute, CERTH, Thessaloniki, Greece
Emails: {charalava, kvotis, pbizopoulos, atriand}@iti.gr
†Department of Cardiology, Transplantation Center, University Medical Center Utrecht, Utrecht, The Netherlands
Email: m.l.handoko@umcutrecht.nl
‡Cardiology Department, Vall d'Hebron University Hospital, Vall d'Hebron Institut de Recerca, Barcelona, Spain
Email: eduard.rodenas@gmail.com
§Institute of Health Informatics, UCL; Amsterdam UMC; NIHR UCLH BRC, London and Amsterdam, UK
Email: f.w.asselbergs@amsterdamumc.nl

***Abstract*—Advancements in GPU-accelerated web-based 3D rendering have enabled high-quality, interactive medical visualization directly within standard browsers, eliminating the need for native software installations. Building on this progress, we present a fully browser-native isosurface renderer designed for volumetric medical datasets. The system supports both trilinear and tricubic interpolation, enabling sub-voxel accurate surface reconstruction through analytic root-finding techniques. To improve rendering efficiency, we extend empty-space skipping strategies to accommodate tricubic interpolation. Additionally, we introduce a novel early rejection test based on Bernstein polynomials, which efficiently identifies and excludes regions devoid of isosurface intersections—yielding up to $1.5\times$ speedups, especially in the computationally intensive tricubic case. While each component exists in native graphics pipelines, their integration into a browser-based framework bridges the gap between desktop-level performance and installation-free access. We validate our approach using contrast-enhanced cardiac CT volumes, achieving interactive frame rates ranging from 25 to 45 frames per second (FPS). on consumer-grade hardware. In a targeted usability study, 7 out of 8 clinicians reported that the system improved their ability to visualize patient data and communicate medical findings.**

***Index Terms*—Isosurface rendering, WebGL, analytic ray casting, cardiac imaging, empty space skipping.**

I. Introduction

Three-dimensional (3D) visualization has become integral to modern cardiac imaging, empowering clinicians to intuitively explore complex anatomical structures. High-quality 3D renderings significantly aid in assessing heart chamber morphology, diagnosing congenital abnormalities, and planning interventions such as valve repair and ventricular remodeling [1]–[4].

Among available visualization techniques, *isosurface rendering* is particularly effective for extracting and depicting surfaces of constant intensity from volumetric CT or MRI data. It enables high-fidelity anatomical reconstructions and supports critical clinical applications such as surgical planning and procedural simulation [1], [5], [6]. In cardiac imaging, isosurface rendering facilitates the integration of anatomical and functional datasets—such as CT angiography and perfusion imaging—providing a more comprehensive diagnostic assessment [7].

Recent advances in browser-based rendering technologies, notably WebGL and the emerging WebGPU, have significantly enhanced real-time 3D visualization capabilities on modern web platforms [7]–[9]. These web-native solutions offer crucial advantages: ease of deployment, device-agnostic compatibility, and privacy-preserving local computation.

Several advanced platforms exemplify the growing maturity of web-based medical visualization. *Kitware's VolView* [10] and *Glance*, both built on `vtk.js`, offer robust browser-native volume rendering capabilities. VolView provides clinical-grade features such as cinematic rendering, annotation tools, and DICOM workflow support. Glance offers a lightweight alternative focused on general-purpose scientific visualization. The OHIF Viewer [11] further extends this trend with a modular, open-source framework for radiological image viewing, featuring DICOMweb integration and plugin-based extensibility. While these platforms excel in traditional volume rendering and diagnostic review, they do not explicitly target analytic isosurface reonstruction.

In contrast, our work focuses on delivering *efficient, high-quality isosurface reconstruction*—a rendering approach not commonly addressed in existing web-based systems. We present a fully browser-native solution that performs *analytic isosurface extraction* with sub-voxel precision, extended to the tricubic case through Csébfalvi's fast interpolation filter [12]. Rendering efficiency is maintained using acceleration structures based on distance-map–driven empty-space skipping [13], [14]. This design enables real-time, high-fidelity isosurface visualization entirely within the browser, bridging the gap between desktop-level accuracy and accessible, platform-independent deployment.

This work was supported by the European Union's Horizon Europe Research and Innovation Programme under Grant Agreement No. 101080430 "AI4HF". The authors thank the AI4HF consortium for their support and collaboration.

Our key contributions are:

- A browser-native isosurface renderer built with WebGL, Three.js, and TensorFlow.js, supporting sub-voxel accuracy via analytic intersections under trilinear and tricubic interpolation, and enabling privacy-preserving isosurface visualization at interactive frame rates on standard desktop browsers.
- Extended analytic isosurface intersection to the tricubic case using Csébfalvi's fast interpolation scheme, and introduced a method for efficient reconstruction of the interpolant within each cell.
- Adaptation of GPU-accelerated acceleration structures—including occupancy and Chebyshev distance maps—for empty-space skipping under tricubic interpolation.
- A Bernstein polynomial–based early rejection test for efficiently pruning non-intersecting cells, effective for both trilinear and tricubic interpolation.

These contributions help extend the capabilities of web-based medical visualization by enabling analytically accurate, high-performance isosurface rendering directly in standard desktop browsers—offering a practical alternative to native applications for accessible and precise surface visualization.[1]

II. Methods

A. Interpolation Models

Volumetric data is represented as a scalar field $f(x, y, z)$, discretely sampled on a regular 3D grid and stored as a 3D texture on the GPU. Each voxel contains a scalar value (e.g., intensity or density) and is accessed using normalized coordinates for efficient memory traversal.

To reconstruct the scalar field continuously between these discrete samples, interpolation is performed within individual grid cells. Each cell comprises a cube defined by eight neighboring corner voxels and serves as the local support for interpolation. Given a sample position (x, y, z), we decompose it into integer and fractional components:

$$x = i + \alpha, \quad y = j + \beta, \quad z = k + \gamma$$

where i, j, k are cell indices, and $\alpha, \beta, \gamma \in [0, 1]$ are normalized local coordinates within the cell. This parameterization defines the domain over which interpolation functions are evaluated.

In this work, we consider two interpolation models of increasing quality: *trilinear interpolation*, which is standard and hardware-accelerated, and *tricubic interpolation*, which introduces second-order corrections for improved continuity and surface reconstruction.

1) Trilinear Interpolation: This is the default method for reconstructing scalar values inside each voxel cell due to its simplicity and GPU accelerated hardware support. The interpolated scalar field is computed using a weighted blend of the eight corner voxel values:

$$f_{\text{trilin}}(x, y, z) = \sum_{p=0}^{1} \sum_{q=0}^{1} \sum_{r=0}^{1} f_{ijk}[p, q, r]\, B_p^1(\alpha)\, B_q^1(\beta)\, B_r^1(\gamma), \tag{1}$$

where $B_0^1(a) = 1 - a$ and $B_1^1(a) = a$ are the degree-one Bernstein basis functions, similarly defined for β and γ. The values $f_{ijk}[p, q, r]$ represent the scalar values at corner offsets (p, q, r) within the cell (i, j, k).

Trilinear interpolation yields a continuous (C^0) scalar field and is well-suited for interactive applications. However, its lack of smooth derivatives introduces faceting artifacts, resulting in a visibly segmented and angular isosurface reconstructions.

2) Tricubic Interpolation with Second-Order Corrections: To improve visual smoothness and continuity, we adopt *tricubic interpolation*, following Csébfalvi [12]. This technique augments the trilinear scalar field with quadratic correction terms based on second-order partial derivatives:

$$\begin{aligned} f_{\text{tricub}}(x, y, z) = f_{\text{trilin}}(x, y, z) &+ f_{xx}(x, y, z) \cdot (\alpha^2 - \alpha)/2 \\ &+ f_{yy}(x, y, z) \cdot (\beta^2 - \beta)/2 \\ &+ f_{zz}(x, y, z) \cdot (\gamma^2 - \gamma)/2, \end{aligned} \tag{2}$$

where the second-order derivative functions f_{xx}, f_{yy}, f_{zz} are reconstructed via trilinear interpolation from precomputed derivative textures, generated using central differencing.

For efficiency, these derivatives are packed alongside the scalar value f in a single four-channel 3D texture with the layout $(f_{xx}, f_{yy}, f_{zz}, f)$. This compact representation enables efficient reconstruction using a single trilinear texture fetch [12].

The resulting interpolant is C^1-continuous and substantially reduces faceting artifacts. It produces visually smoother isosurfaces with continuous curvature transitions, achieving quality comparable to Catmull–Rom splines.

B. Analytic Isosurface Intersection

In volume rendering, an *isosurface* represents the set of all points (x, y, z) in a scalar field $f(x, y, z)$ where the field equals a constant target *isovalue* f_{iso}. Formally, the isosurface is defined by the implicit equation:

$$f(x, y, z) = f_{\text{iso}}.$$

To visualize this surface, a *viewing ray* is cast from each image pixel into the volume. A viewing ray originates at the camera position $\mathbf{c}$ and travels in direction $\mathbf{d}$, forming a parametric line given by:

$$\mathbf{r}(t) = \mathbf{c} + t\mathbf{d}, \quad t \geq 0.$$

Our objective is to find the point along this ray where it first intersects the isosurface, i.e., the smallest t for which $f(\mathbf{r}(t)) = f_{\text{iso}}$.

To efficiently traverse the volume, we use a discrete differential analyzer (DDA) [15], which incrementally advances

[1]The full implementation and an interactive demo are available at: `github.com/XaralampidisBasilis/ISO-Viewer`.

the viewing ray through the regular 3D grid in the order of intersected grid cells. This partitions the ray into segments bounded by cell entry and exit points.

Within each segment, we define the local parametric line, which linearly interpolates between the cell's entry $\mathbf{p}_{\text{entry}}$ and exit points $\mathbf{p}_{\text{exit}}$:

$$\mathbf{p}(\tau) = \mathbf{p}_{\text{entry}}(1-\tau) + \mathbf{p}_{\text{exit}}\tau, \quad \tau \in [0,1] \tag{3}$$

Substituting the local parametric line into the interpolated scalar field yields a one-dimensional interpolation function $f(\mathbf{p}(\tau))$, which describes scalar variation along the ray segment within the cell. To evaluate whether the ray intersects the isosurface, we subtract the isovalue to construct the isosurface residual function:

$$\rho_{\text{iso}}(\tau) = f(\mathbf{p}(\tau)) - f_{\text{iso}}, \quad \tau \in [0,1] \tag{4}$$

The intersection test then reduces to determining whether the isosurface residual function $\rho_{\text{iso}}(\tau)$ has at least one real root within the normalized interval $\tau \in [0,1]$, defined over the cell. This function takes the form of a univariate polynomial, with its degree determined by the interpolation model—cubic for trilinear, and quintic for tricubic.

When a root is detected within the cell, traversal is terminated, and the smallest valid root $\tau_{\text{iso}} \in [0,1]$ is computed analytically using the GPU polynomial solver by Peters [16], based on Yuksel's interval arithmetic [17]. The corresponding isosurface intersection point is then recovered by substituting τ_{iso} into the local ray equation: $\mathbf{p}_{\text{iso}} = \mathbf{p}(\tau_{\text{iso}})$.

Efficient evaluation of the residual function $\rho_{\text{iso}}(\tau)$ and detection of real roots within the normalized interval $\tau \in [0,1]$ are central to the accuracy and performance of analytic isosurface intersection, as they represent the most computationally expensive operations in the rendering pipeline. The following sections detail these processes for both trilinear and tricubic interpolation models.

1) Trilinear Intersections: With trilinear interpolation, the residual function becomes a cubic polynomial [18]–[20]:

$$\rho_{\text{iso}}(\tau) = \sum_{n=0}^{3} c_n \tau^n, \quad \tau \in [0,1]. \tag{5}$$

The coefficients $\mathbf{c} = [c_0, c_1, c_2, c_3]^\top$ are computed by evaluating the trilinearly interpolated scalar field at four equidistant positions along the local parametric line (3), inside the cell, and solving the resulting cubic Vandermonde system:

$$\mathbf{c} = \mathbf{V}_{\text{cubic}} \boldsymbol{\rho}_{\text{iso}},$$

where $\mathbf{V}_{\text{cubic}} \in \mathbb{R}^{4\times4}$ is the inverse Vandermonde matrix associated with the four sampling positions, and $\boldsymbol{\rho}_{\text{iso}} \in \mathbb{R}^{4\times1}$ contains the corresponding isosurface residual values. Efficiency is improved by reusing one sample across adjacent cells and globally caching the inverse Vandermonde matrix to avoid redundant computation.

To effectively detect whether an intersection occurs, we compute the critical points of the residual function $\rho_{\text{iso}}(\tau)$ within the interval $\tau \in [0,1]$. These are obtained by solving the derivative equation:

$$\frac{d\rho_{\text{iso}}}{d\tau} = 0, \quad \tau \in [0,1]$$

which reduces to quadratic equation that can be solved efficiently and robustly. We then examine whether a sign change exists between the function values at these critical points and the previously evaluated equidistant samples. A sign change confirms the presence of a real root—and thus an isosurface intersection—within the interval.

2) Tricubic Intersections: The tricubic case follows an analogous structure to the trilinear formulation but operates on a higher-order residual function. Here, the isosurface residual $\rho_{\text{iso}}(\tau)$ becomes a quintic polynomial:

$$\rho_{\text{iso}}(\tau) = \sum_{n=0}^{5} c_n \tau^n, \quad \tau \in [0,1].$$

While a general quintic polynomial requires six values to determine its coefficients, we introduce a more compact formulation that uses only four. This reduction leverages the specific structure of the tricubic interpolation equation (2), which consists of four trilinear interpolants $f_{xx}, f_{yy}, f_{zz}, f$ augmented by three spatial quadratic correction terms. The coefficient computation can then be expressed in the following factored matrix form:

$$\mathbf{C} = \mathbf{V}_{\text{cubic}}\, \mathbf{R}_{\text{iso}}\, \mathbf{V}_{\text{quad}}^\top,$$

where $\mathbf{C} \in \mathbb{R}^{4\times3}$ is the intermediate coefficients matrix, and $\mathbf{R}_{\text{iso}} \in \mathbb{R}^{4\times3}$ contains the intermediate residual values. The matrices $\mathbf{V}_{\text{cubic}} \in \mathbb{R}^{4\times4}$ and $\mathbf{V}_{\text{quad}} \in \mathbb{R}^{3\times3}$ are cached, inverse Vandermonde matrices corresponding to the cubic and quadratic sampling positions, respectively. To construct the intermediate residuals matrix, we evaluate:

$$\mathbf{R}_{\text{iso}} = \mathbf{F}^\top \mathbf{G} - \mathbf{J} f_{\text{iso}},$$

where $\mathbf{F} \in \mathbb{R}^{4\times4}$ is composed of column vectors $\mathbf{F}_i$, each representing a packed trilinear texture sample evaluated at one of four equidistant positions $\mathbf{p}(\tau_i)$ along the ray:

$$\mathbf{F}_i = \left[f_{xx}(\mathbf{p}(\tau_i)),\ f_{yy}(\mathbf{p}(\tau_i)),\ f_{zz}(\mathbf{p}(\tau_i)),\ f(\mathbf{p}(\tau_i))\right]^\top.$$

The matrix $\mathbf{G} \in \mathbb{R}^{4\times3}$ encodes the quadratic correction terms, evaluated at three distinct positions, with each column taking the form:

$$\mathbf{G}_i = \left[(\alpha_i^2 - \alpha_i)/2, (\beta_i^2 - \beta_i)/2, (\gamma_i^2 - \gamma_i)/2, 1\right]^\top.$$

The matrix $\mathbf{J} \in \mathbb{R}^{4\times3}$ denotes a matrix of ones. Finally, the quintic coefficients $\mathbf{c} = [c_0, \ldots, c_5]^\top$ are extracted by summing the anti-diagonals of the intermediate coefficients matrix $\mathbf{C}$. As in the trilinear case efficiency is improved by reusing one packed sample across adjacent cells and globally caching the inverse Vandermonde matrices.

Given the coefficients, intersection detection proceeds by efficiently evaluating the quintic residuals at 16 equidistant positions, within $\tau \in [0,1]$, in parallel batches of four, using

Horner's method . This enables fast sign-change detection and robust identification of real roots—and thus isosurface intersections—within the interval.

3) Bernstein Early Rejection: To reduce the computational overhead of intersection detection, we introduce the *Bernstein Early Rejection (BER)* test—a fast root exclusion method that expresses the isosurface residual polynomial in the Bernstein basis over the ray segment within a cell:

$$\rho_{\text{iso}}(\tau) = \sum_{n=0}^{d} b_n \, B_n^d(\tau), \quad \tau \in [0,1],$$

where $B_n^d(\tau) = \binom{d}{n}\tau^n(1-\tau)^{d-n}$ for $n = 0, ..., d$ are the Bernstein basis functions of degree d, and b_n are the associated Bernstein coefficients. Here, $d = 3$ or 5 corresponds to trilinear and tricubic interpolation, respectively.

By the convex hull property of Bernstein polynomials, if all b_n share the same sign, the polynomial cannot cross zero in the interval $\tau \in [0,1]$, and the cell is conclusively rejected. If a sign change is detected, we proceed with the full coefficient reconstruction and perform the standard intersection test.

The Bernstein coefficients b_n are efficiently computed from the residual samples using precomputed transformation matrices. These matrices combine inverse Vandermonde systems with basis conversion operators that transform polynomials from the power basis to the Bernstein basis. This formulation ensures minimal computational overhead while improving numerical stability. Specifically, for the trilinear case, the Bernstein coefficients are computed as:

$$\mathbf{b} = \mathbf{W}_{\text{cubic}} \boldsymbol{\rho}_{\text{iso}},$$

where $\mathbf{W}_{\text{cubic}} \in \mathbb{R}^{4\times4}$ is the precomputed conversion matrix to cubic Bernstein basis from residual values. For the tricubic case, the intermediate Bernstein coefficient matrix is given by:

$$\mathbf{B} = \left(\mathbf{W}_{\text{cubic}} \, \mathbf{R}_{\text{iso}} \, \mathbf{W}_{\text{quad}}^{\top}\right) \circ \mathbf{M}_{\text{quint}},$$

where $\mathbf{W}_{\text{quad}} \in \mathbb{R}^{3\times4}$ and $\mathbf{W}_{\text{cubic}} \in \mathbb{R}^{4\times4}$ are the precomputed conversion matrices to the quadratic and cubic Bernstein basis, respectively. The matrix $\mathbf{M}_{\text{quint}}\mathbb{R}^{4\times3}$ transforms products of lower-order Bernstein basis functions, cubic and quadratic, into the equivalent quintic Bernstein basis. The Hadamard product $\circ$ is applied element-wise to extract the final coefficients. This matrix corresponds to the Bernstein basis product identity:

$$B_n^3(t) \cdot B_m^2(t) = \frac{\binom{3}{n}\binom{2}{m}}{\binom{5}{n+m}} \cdot B_{n+m}^5(t).$$

Again, the quintic bernstein coefficients $\mathbf{b} = [b_0, \ldots, b_5]^{\top}$ are extracted by summing the anti-diagonals of the intermediate coefficient matrix B.

C. Empty Space Skipping

To reduce unnecessary computation in sparse volumes, we implement empty space skipping techniques from prior work [13], [14], [21], by leveraging three GPU-accelerated data structures: the *extrema map*, the *occupancy map*, and the *Chebyshev distance map*. These structures guide rays toward regions likely to contain the isosurface, reducing unnecessary evaluations and minimizing memory overhead.

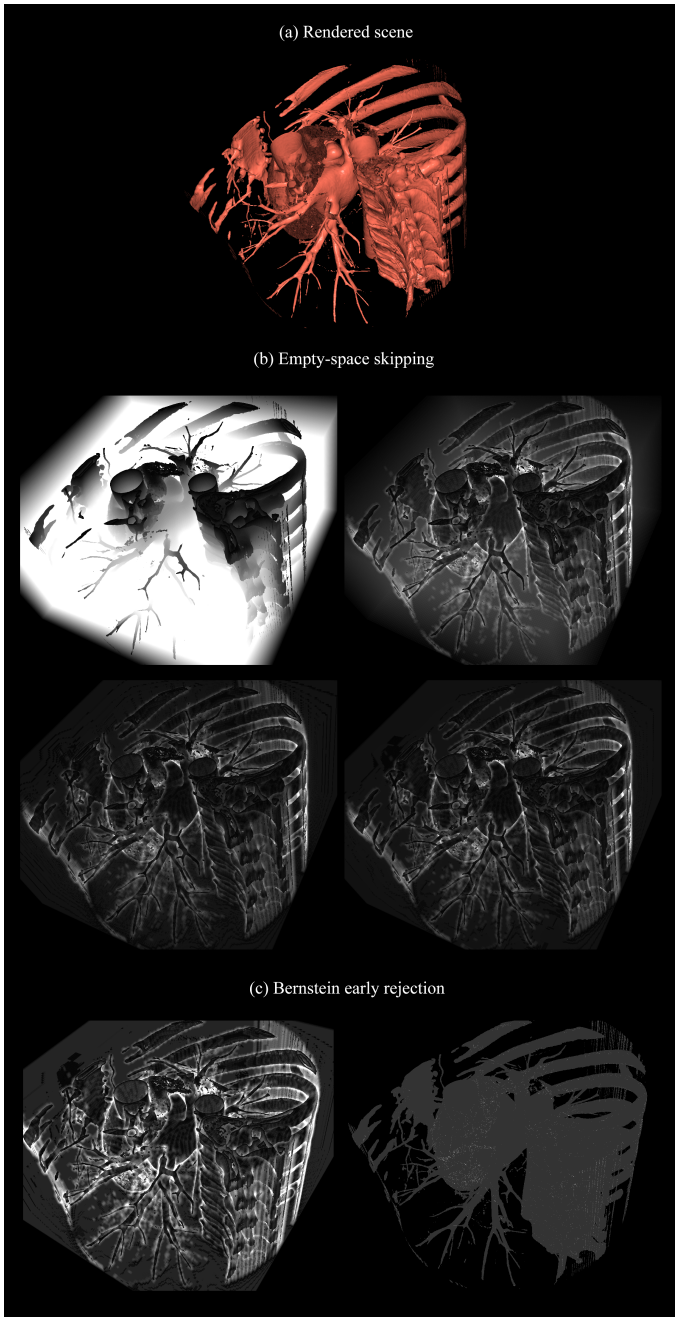

Fig. 1. (a) Rendered scene with tricubic interpolation and constant block size $B = 3$ (b) Relative texture fetch count across different empty space skipping strategies. Brighter regions indicate higher fetch counts. (Top-left) Baseline rendering with no skipping. (Top-right) Occupancy map acceleration. (Bottom-left) Isotropic distance map. (Bottom-right) Anisotropic and extended anisotropic distance maps, which produce visually indistinguishable results. c) Intersection test counts with and without Bernstein early rejection, using anisotropic distance-based skipping. (Left) Without early rejection—brighter regions indicate counts approaching 50. (Right) With Bernstein rejection—counts are significantly reduced; to visualize them, the maximum value is rescaled to 5.

The extrema map stores coarse scalar bounds for each B^3-

sized block of cells. These bounds are derived by aggregating the local extrema computed per cell, where the extremum behavior is determined by the underlying interpolation model, trilinear or tricubic.

1) Trilinear Cell Extrema: Leveraging the convex hull property of the trivariate Bernstein basis (1), the extrema of a trilinear cell are given directly by the minimum and maximum of its eight corner voxel values.

2) Tricubic Cell Extrema: Unlike trilinear interpolation, the tricubic interpolant is non-convex (2), meaning its extrema do not necessarily lie at the voxel corners of the cell. Direct computation of these extrema is intractable due to the polynomial's complexity. Instead, we approximate the bounds by leveraging the convex hull property of the trivariate Bernstein basis. Reformulating the interpolation in this basis ensures that the extrema of the interpolated function are tightly bounded by the minimum and maximum Bernstein coefficients.

The trivariate Bernstein representation of the tricubic interpolant is given by:

$$f_{\text{tricub}}(x,y,z) = \sum_{u=0}^{3}\sum_{v=0}^{3}\sum_{w=0}^{3} b_{uvw}\, B_u^3(\alpha)\, B_v^3(\beta)\, B_w^3(\gamma), \quad (6)$$

where $B_u^3(\alpha) = \binom{3}{u}\alpha^u(1-\alpha)^{3-u}$ are cubic Bernstein basis functions, and likewise for β and γ. The coefficients b_{uvw} are derived from the scalar field and second-order derivatives stored at voxel corners using:

$$b_{uvw} = \sum_{p=0}^{1}\sum_{q=0}^{1}\sum_{r=0}^{1} h_{uvw}[p,q,r]\cdot W[p,u]\cdot W[q,v]\cdot W[r,w], \quad (7)$$

with the intermediate terms defined as:

$$\begin{aligned} h_{uvw}[p,q,r] = f[p,q,r] &+ f_{xx}[p,q,r]\cdot M[p,u] \\ &+ f_{yy}[p,q,r]\cdot M[q,v] \\ &+ f_{zz}[p,q,r]\cdot M[r,w], \end{aligned}$$

Here, $\mathbf{W} \in \mathbb{R}^{2\times 4}$ is the elevation matrix from degree-one to degree-three Bernstein basis, and $\mathbf{M} \in \mathbb{R}^{2\times 4}$ encodes the contributions of the quadratic correction terms. This formulation enables efficient estimation of cell extrema directly from the packed precomputed 3D texture used in tricubic interpolation.

A binary occupancy map is derived from these extrema bounds: a block is marked as occupied if the isovalue lies within its scalar range, guaranteeing the potential presence of an isosurface via the Intermediate Value Theorem, allowing efficient culling based on the isovalue.

To accelerate traversal, Chebyshev distance maps are computed, from the occupancy map, to allow rays to skip multiple empty blocks in a single step for given directions. We use three variants: isotropic (single scalar per block), anisotropic (per-octant distances), and extended anisotropic (per-axis octant distances-encoded using a 16-bit layout), offering a trade-off between memory usage and culling precision.

During rendering, rays alternate between coarse block skipping and fine-grained cell traversal [22]. In the block stage, the ray samples the distance map to leap over empty regions. Upon reaching an occupied block, it switches to cell-level traversal, performing precise isosurface intersection tests. This hierarchical scheme significantly reduces redundant evaluation, improves memory coherence, and enhances GPU efficiency.

D. Derivative Reconstruction

Accurate gradient and Hessian reconstruction (Fig 2) improves shading fidelity, enhancing visual perception in isosurface rendering. These derivatives are also a prerequisite for differential surface analysis. Following Kindlmann et al. [23], [24], principal curvatures can be derived from the gradient and Hessian, enabling advanced effects such as curvature-guided shading, though such techniques are beyond the scope of this paper.

1) Gradient: To compute the gradient $\mathbf{g}$ at the isosurface intersection, we adopt the triquadratic B-spline interpolation method proposed by Csébfalvi [25]. This approach yields smooth, analytically derived partial derivatives that are C^1-continuous and requires only eight trilinear texture fetches, making it practical for real-time rendering.

2) Hessian: The same filtering scheme [25] can be used to estimate the mixed second-order partial derivatives forming the off-diagonal elements of the Hessian matrix H, by reusing the same eight trilinear fetches from the gradient estimation. For pure second-order derivatives, this method is not applicable, as explained in [26]. Instead, for the trilinear case, we compute them on-the-fly using central differencing evaluated at the isosurface intersection point. In the tricubic case, second-order derivatives are obtained directly by sampling the precomputed 3D texture that stores the packed second-order terms.

E. Shading

Once the intersection point on the isosurface is identified and we generate the gradients, we apply an illumination model to approximate the reflected light intensity. We use the Blinn-Phong local illumination model, which expresses the outgoing light as a combination of ambient, diffuse, and specular components. The illumination is computed as:

$$I = k_a I_a + k_d I_l \max(\mathbf{n}\cdot\mathbf{l}, 0) + k_s I_l \max(\mathbf{n}\cdot\mathbf{h}, 0)^s$$

where $\mathbf{n}$ is the surface normal, $\mathbf{l}$ the light vector, $\mathbf{v}$ the view vector, and $\mathbf{h}$ the halfway vector. The coefficients k_a, k_d, and k_s control ambient, diffuse, and specular reflectance, with exponent s controlling highlight sharpness.

The surface normal is estimated from the normalized gradient of the volume at the intersection point. Since gradients are unreliable in homogeneous regions, we suppress diffuse and specular lighting contributions where the gradient magnitude falls below a defined threshold [27], effectively enhancing contrast around anatomically relevant structures.

In addition, to improve shape perception, we darken regions where the surface normal aligns with the view direction, emphasizing silhouette edges. This accentuates anatomical features and boundaries, making the visualization more interpretable in clinical and research contexts.

F. Implementation

The system is built entirely within the Three.js framework, utilizing its scene graph, camera control, and shader infrastructure to enable fully interactive 3D rendering in the browser. GPU-accelerated preprocessing is carried out using TensorFlow.js [28], which enables efficient in-browser computation through its WebGL backend. Custom compute shaders are implemented using TensorFlow.js's `GPGPUProgram` interface.

The input volume is first resized, normalized, and encoded as a 3D tensor. From this, an interpolation map is generated by computing and storing second-order partial derivatives, along with the volume values. A custom compute shader is then used to construct the extrema map, supporting both trilinear and tricubic interpolation. Based on the extrema, binary occupancy maps are created through thresholding relative to the target isovalue. Finally, distance maps are derived from the occupancy map using custom compute shaders adapted from prior work [14].

To ensure responsiveness, the system caches the extrema map tensor and regenerates dependent data structures—such as occupancy and distance maps—on demand during isovalue updates. Final rendering is performed using GLSL shaders integrated via `THREE.ShaderMaterial`, enabling real-time, high-fidelity visualization entirely within the browser environment, using WebGL.

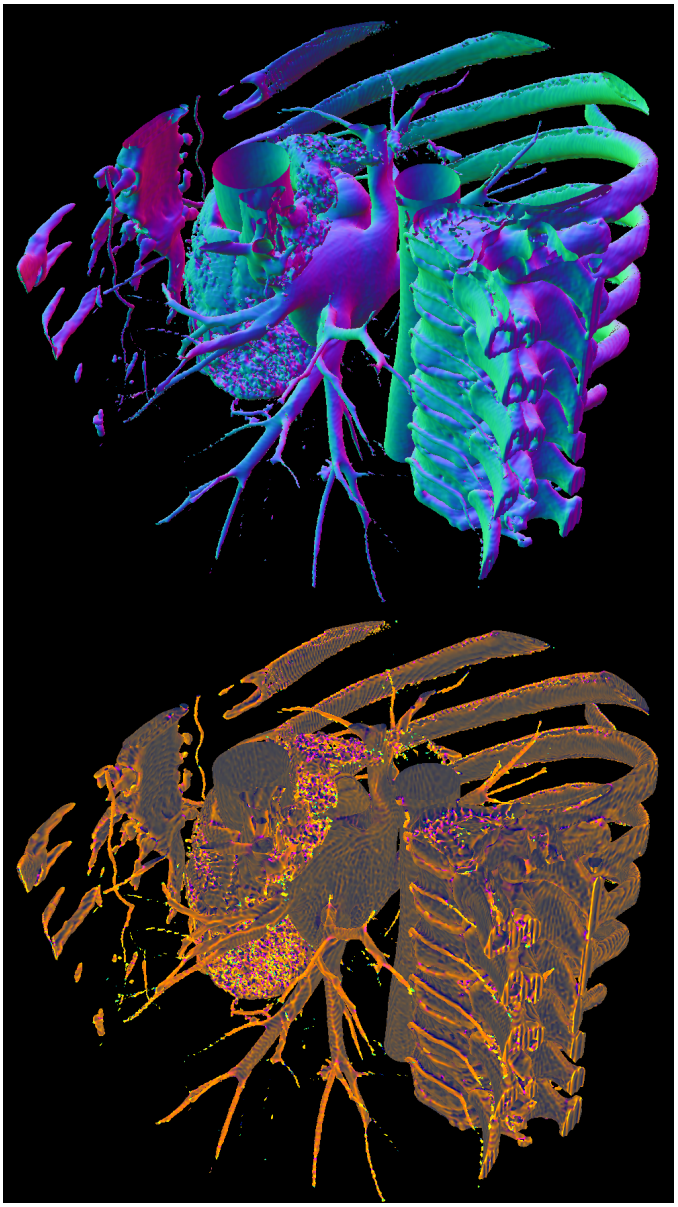

Fig. 2. Normals and principal curvatures visualization on a cardiac CT isosurface. (Top) Surface normals. (Bottom) Principal curvature classification: gray for planar regions, cyan for elliptic concave surfaces (both curvatures negative), gold for elliptic convex surfaces (both curvatures positive), magenta for saddle points (opposing curvature signs), blue for parabolic concave (one zero, one negative), and orange for parabolic convex (one zero, one positive).

III. Results

All experiments were performed on a standard laptop equipped with integrated Intel Iris Xe Graphics. Tests were executed in Google Chrome version 125 with WebGL2 enabled, using a screen resolution of 1920×911 pixels. For benchmarking, we employed cardiac CT volumes from publicly available datasets [29], [30]. The volumes were downsampled to a maximum resolution of 320^3 voxels to retain anatomical accuracy while reducing memory requirements, as the original full-resolution volumes were memory-intensive. Peak GPU memory usage was below 4 GB during preprocessing and initialization while interactive rendering stabilized near 1 GB.

We evaluated both trilinear and tricubic interpolation models across multiple acceleration strategies: no skipping, Occupancy Map (OM), Isotropic Distance Map (Iso. DM), Anisotropic Chebyshev Distance Map (Aniso. DM), and Extended Anisotropic Distance Map (Ext. DM). Each setup was tested with and without the Bernstein-Based Early Rejection (BER) test. To ensure consistency and fairness, all measurements were collected under identical viewing parameters with a fixed block size of $B = 3$. Fig 1b illustrates the texture fetch density across different strategies. A summary of the average frame rates (FPS) achieved for each configuration is provided in Table I.

To assess preprocessing overhead, we measured the construction time of each structure after an isovalue update. The occupancy map (OM) was fastest ($< 120\,\mathrm{ms}$), followed by the isotropic (210 ms), anisotropic (250 ms), and extended anisotropic distance maps (480 ms). These update latencies are infrequent and outweighed by the system's sustained interactivity. The coarse extrema map, computed once per volume, took 160 ms for trilinear and 240 ms for tricubic interpolation.

A. Discussion

Qualitatively, tricubic interpolation significantly enhances surface appearance, especially in vascular regions of contrast-enhanced cardiac datasets (Fig. 3). By producing more precise isosurface intersections, it reduces aliasing and geometric faceting, resulting in smoother silhouettes. In contrast, trilinear

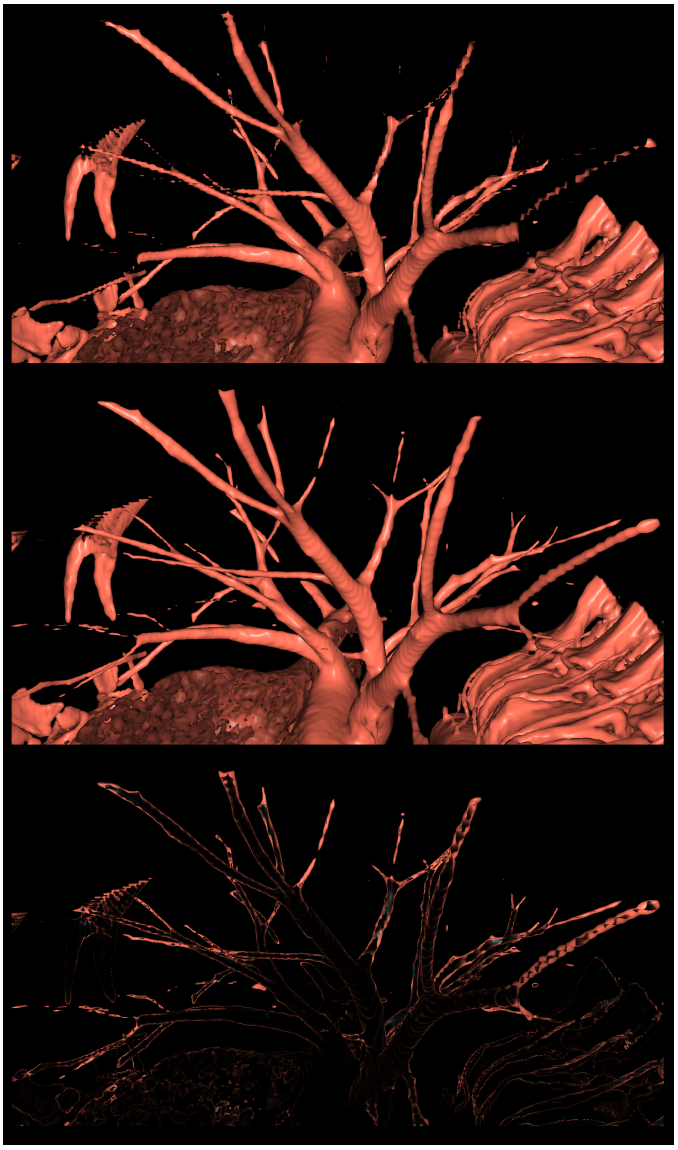

Fig. 3. Analytic isosurface renderings of a contrast-enhanced cardiac CT volume: (a) Trilinear interpolation, (b) Tricubic interpolation with second-order correction as proposed by Csébfalvi [12], and (c) Absolute difference between the two methods. Tricubic reconstruction yields smoother surfaces and improved continuity compared to the faceted artifacts observed in the trilinear result.

interpolation exhibits noticeable angular distortions and less faithful surface continuity. The bottom row of Fig. 3 visualizes absolute differences between methods, illustrating the local geometric improvements achieved by the higher-order interpolation.

All evaluated acceleration strategies demonstrated substantial reductions in computational load while preserving visual fidelity (Fig. 1b). Among them, the Anisotropic Distance Map (Aniso. DM) achieved the most effective trade-off between memory usage and performance, exhibiting frame rates nearly identical to the more complex Extended Distance Map (Ext. DM) for the tested volume.

The Bernstein-Based Early Rejection (BER) technique consistently delivered additional performance gains across all configurations (Table. I). Its impact was particularly pronounced in the tricubic interpolation setting, where it significantly reduced the cost associated with finding roots of quintic polynomials (Fig. 1c).

TABLE I
AVERAGE FRAME RATES (FPS) ACROSS INTERPOLATION AND ACCELERATION STRATEGIES

Interpolation	BER	Base	OM	Iso. DM	Aniso. DM	Ext. DM
Trilinear	No	18	34	45	47	47
	Yes	25	38	49	50	50
Tricubic	No	6	14	17	19	19
	Yes	13	21	26	30	31

B. Survey

To assess the usability, clinical relevance, and perceived value of our browser-based cardiac imaging system, we conducted a structured survey involving eight healthcare professionals from multiple countries. Participants included cardiologists, imaging researchers, and technical physicians, representing a diverse range of expertise within the medical imaging field. The survey covered demographics, technology usage habits, and perceptions of the system's clinical utility.

1) Baseline Characteristics: Respondents ranged in age from 31 to 54 years, with the most common ages being 44 and 45. The group was predominantly male and consisted primarily of practicing cardiologists, alongside technical physicians and researchers. Clinical experience varied from 6 to over 25 years, reflecting a balanced distribution across mid- and late-career professionals.

2) Technology Usage: All participants reported a strong familiarity with browser-based systems. Most indicated regular use of digital imaging tools such as CVI, OsiriX, and RadiAnt DICOM Viewer, suggesting that the user group was well-acquainted with modern medical imaging workflows—an encouraging context for evaluating browser-native alternatives.

3) Tool Acceptance: Overall, participant feedback reflected strong acceptance of the system. Seven of the eight respondents agreed that the tool would improve their ability to visualize patient data—three strongly agreed, and four slightly agreed. The same group also indicated that the system could facilitate clearer communication with patients. Regarding ease of use, all eight participants expressed agreement that the tool would be easy to learn and operate, with six strongly agreeing and two slightly agreeing. Notably, no negative responses were recorded across any of the usability or clinical relevance questions, underscoring consistent positive sentiment across the surveyed group.

IV. FUTURE WORK

A key direction for future work is extending support to large volumetric datasets (e.g., 512^3 voxels). While memory efficiency was not the central focus of this study, it can be significantly enhanced using techniques such as brick-based streaming and optimized texture memory management [24].

Tricubic interpolation, in particular, increases memory demands by a factor of four, motivating further optimization.

Additionally, future work will focus on minimizing GPU-to-CPU-to-GPU data transfers between TensorFlow.js and the Three.js rendering pipeline. Reducing this overhead is crucial for maintaining real-time performance and improving scalability, especially when working with larger volumetric datasets.

V. Conclusion

We introduced a fully browser-based isosurface renderer that integrates analytic intersections, higher-order interpolation, and empty-space skipping to support high-fidelity, interactive visualization of medical volumetric data. Our implementation incorporates Csébfalvi's tricubic interpolation filter within a WebGL rendering pipeline and leverages Chebyshev distance maps to accelerate traversal, enabling sub-voxel surface accuracy and real-time performance using only standard web technologies.

In contrast to existing web-based platforms that primarily focus on general-purpose volume rendering, our system is specifically designed for accurate isosurface reconstruction. Through the combination of analytic methods and GPU-efficient acceleration structures, it delivers smooth, curvature-preserving surfaces with interactive frame rates—even on low end hardware—making it a practical and accessible solution for clinical and research use cases.

A usability survey with eight clinicians indicated strong acceptance, with most participants reporting the tool as useful for both diagnostic visualization and patient communication. The entirely client-side implementation further supports data privacy, which is essential in clinical environments.

In summary, our method extends the capabilities of web-based medical visualization, offering a viable alternative to native applications for isosurface-centric tasks. Future work will explore broader clinical validation and support for additional features provided from cardiologists.

References

[1] J. O. Bescos, J. Smit, and C. H. Slump, "Interactive 4d cardiac mri imaging based on iso-surface volume rendering," in *Medical Imaging 2002: Visualization, Image-Guided Procedures, and Display*, vol. 4681. SPIE, 2002, pp. 322–329.

[2] F. Yang, W. Zuo, K. Wang, and H. Zhang, "Visualization of segmented cardiac anatomy with accelerated rendering method," in *2009 36th Annual Computers in Cardiology Conference (CinC)*. IEEE, 2009, pp. 789–792.

[3] Q. Zhang, R. Eagleson, and T. M. Peters, "Gpu-based visualization and synchronization of 4-d cardiac mr and ultrasound images," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 5, pp. 878–890, 2012.

[4] Z. Wang, R. Yi, X. Wen, C. Zhu, and K. Xu, "Cardiovascular medical image and analysis based on 3d vision: A comprehensive survey," *Meta-Radiology*, p. 100102, 2024.

[5] S. J. Kim, B. I. Choi, S. H. Kim, and J. Y. Lee, "Three-dimensional imaging for hepatobiliary and pancreatic diseases: emphasis on clinical utility," *Indian Journal of Radiology and Imaging*, vol. 19, no. 01, pp. 7–15, 2009.

[6] P. A. Nugroho, D. K. Basuki, and R. Sigit, "3d heart image reconstruction and visualization with marching cubes algorithm," in *2016 International Conference on Knowledge Creation and Intelligent Computing (KCIC)*. IEEE, 2016, pp. 35–41.

[7] R. Johnson *et al.*, "A review of three-dimensional medical image visualization," *Health Data Science*, no. 1, pp. 83–101, 2022.

[8] L. Franke and D. Haehn, "Modern scientific visualizations on the web," in *Informatics*, vol. 7, no. 4. MDPI, 2020, p. 37.

[9] C. Bohak, Ž. Lesar, P. Lavric, and M. Marolt, "Web-based 3d visualisation of biological and medical data," *Biomedical Visualisation: Volume 6*, pp. 1–18, 2020.

[10] F. Li, C. Harris, S. R. Aylward, D. Lazzeri, P. Elliott, S. Jadhav, T. Suchel, T. Thirion, and S. Jhaveri, "Kitwaremedical/volview: Volview 4.0," Nov. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7328066

[11] "Open Health Imaging Foundation — ohif.org," https://ohif.org/, [Accessed 09-01-2025].

[12] B. Csébfalvi, "Beyond trilinear interpolation: Higher quality for free," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–8, 2019.

[13] L. Deakin and M. Knackstedt, "Accelerated volume rendering with chebyshev distance maps," in *SIGGRAPH Asia 2019 Technical Briefs*, 2019, pp. 25–28.

[14] ——, "Efficient ray casting of volumetric images using distance maps for empty space skipping. comput. vis. media 6, 53–63," 2020.

[15] J. Amanatides, A. Woo *et al.*, "A fast voxel traversal algorithm for ray tracing." in *Eurographics*, vol. 87, no. 3, 1987, pp. 3–10.

[16] C. Peters, T. Patel, W. Usher, and C. R. Johnson, "Ray tracing spherical harmonics glyphs." in *VMV*, 2023, pp. 21–31.

[17] C. Yuksel, "A fast & robust solution for cubic & higher-order polynomials," in *ACM SIGGRAPH 2022 Talks*, 2022, pp. 1–2.

[18] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan, "Interactive ray tracing for isosurface rendering," in *Proceedings Visualization'98 (Cat. No. 98CB36276)*. IEEE, 1998, pp. 233–238.

[19] G. Marmitt, A. Kleer, I. Wald, H. Friedrich, and P. Slusallek, "Fast and accurate ray-voxel intersection techniques for iso-surface ray tracing." in *VMV*, vol. 4, 2004, pp. 429–435.

[20] G. Marmitt, H. Friedrich, and P. Slusallek, "Interactive volume rendering with ray tracing." in *Eurographics (State of the Art Reports)*, 2006, pp. 115–136.

[21] A. Es and V. İşler, "Accelerated regular grid traversals using extended anisotropic chessboard distance fields on a parallel stream processor," *Journal of Parallel and Distributed Computing*, vol. 67, no. 11, pp. 1201–1217, 2007.

[22] M. Sramek and A. Kaufman, "Fast ray-tracing of rectilinear volume data using distance transforms," *IEEE Transactions on visualization and computer graphics*, vol. 6, no. 3, pp. 236–252, 2002.

[23] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller, "Curvature-based transfer functions for direct volume rendering: Methods and applications," in *IEEE Visualization, 2003. VIS 2003.* IEEE, 2003, pp. 513–520.

[24] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. H. Gross, "Real-time ray-casting and advanced shading of discrete isosurfaces," in *Computer graphics forum*, vol. 24, no. 3. Citeseer, 2005, pp. 303–312.

[25] B. Csébfalvi, "One step further beyond trilinear interpolation and central differences: Triquadratic reconstruction and its analytic derivatives at the cost of one additional texture fetch," in *Computer Graphics Forum*, vol. 42, no. 2. Wiley Online Library, 2023, pp. 191–200.

[26] C. Sigg and M. Hadwiger, "Fast third-order texture filtering," in *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, M. Pharr, Ed. Boston, MA, USA: Addison-Wesley, 2005, ch. 20, pp. 313–329.

[27] M. Ikits, J. Kniss, A. Lefohn, and C. Hansen, "Volume rendering techniques," in *GPU Gems*, R. Fernando, Ed. Addison-Wesley, 2004, ch. 39, pp. 667–692. [Online]. Available: https://developer.nvidia.com/gpugems/gpugems/part-vi-beyond-triangles/chapter-39-volume-rendering-techniques

[28] D. Smilkov, N. Thorat, Y. Assogba, C. Nicholson, N. Kreeger, P. Yu, S. Cai, E. Nielsen, D. Soegel, S. Bileschi *et al.*, "Tensorflow. js: Machine learning for the web and beyond," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 309–321, 2019.

[29] X. Zhuang, "Multivariate mixture model for myocardial segmentation combining multi-source images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 12, pp. 2933–2946, 2018.

[30] X. Zhuang and J. Shen, "Multi-scale patch and multi-modality atlases for whole heart segmentation of mri," *Medical image analysis*, vol. 31, pp. 77–87, 2016.