2026 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW) | 979-8-3195-0529-3/26/$31.00 ©2026 IEEE | DOI: 10.1109/VRW70859.2026.00175

Silhouette Sculptor: A Silhouette-Based Volume and 3D Mesh Refinement

Goksel Tokur*
MS in CS
Waukesha, WI, USA

Simon Koch†
Medical AI Institute
Col. of Osteopathic
Medicine, SHSU
Conroe, TX, USA

Doga Demirel‡
School of Computer Science
University of Oklahoma
Norman, OK, USA

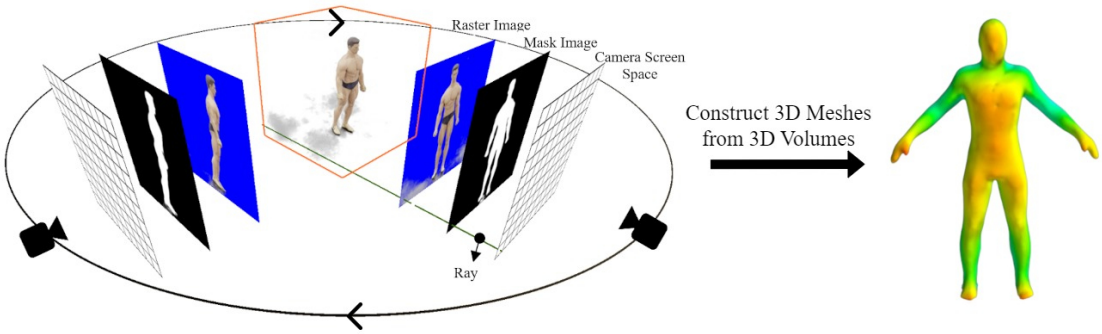

Figure 1: Overview of the Silhouette Sculptor.

Abstract

We introduce an innovative method called *Silhouette Sculptor*, which utilizes an object's silhouette to extract hole-free, accurate 3D meshes from volumetric data generated by Neural Radiance Fields (NeRF). Silhouette Sculptor leverages NeRF alongside a machine learning model specialized in silhouette segmentation to improve accuracy. NeRF has recently gained significant attention for representing complex scenes using continuous volumetric representations. However, reconstructing accurate 3D meshes from NeRF volumes remains challenging due to defects such as surface holes, noise, and inaccuracies introduced during volumetric scene construction and rendering.

Our approach addresses these challenges by incorporating stenciling, volume refinement, and mesh refinement stages into the reconstruction pipeline. We use machine learning models trained to accurately segment object silhouettes, particularly for humans, from rendered 2D raster images of the volumetric scene. By refining defects and integrating 2D silhouette information from multiple viewing angles into the 3D volumetric rendering process, we improve volumetric rendering quality and the accuracy of the extracted 3D meshes.

Index Terms: 3D Mesh Reconstruction; Neural Radiance Fields; Volumetric Rendering.

1 Introduction

3D modeling and rendering techniques, including volumetric rendering, have a wide range of practical applications across industries such as media [5], video games [23], industrial manufacturing, and medical imaging [6]. Polygonal meshes are the most widely used representation of 3D objects in computer graphics applications. This preference creates a growing demand for effective and automated reconstruction of realistic 3D meshes. Rapid advances in modern graphics hardware and recent innovations in computer graphics, including artificial intelligence and neural rendering, enable techniques for reconstructing 3D models from collections of images captured from multiple viewpoints.

One promising technique for generating 3D object representations is Neural Radiance Fields (NeRF) [16], which has gained considerable interest due to its ability to represent complex 3D scenes as continuous volumetric fields. NeRF offers significant improvements over earlier techniques such as Structure from Motion (SfM) [9], Multi-view Stereo as an extension of SfM, and visual hulls [10]. While NeRF is regarded as a state-of-the-art approach in its domain, accurately extracting 3D meshes from NeRF volumetric data remains challenging due to defects, unwanted noise, and inaccuracies that may arise during volumetric scene construction and rendering.

To address these challenges, we propose a novel multi-stage pipeline; an overview and representative reconstruction result are shown in Fig. 1. The main stages are:

1. Construct a volumetric representation of a scene containing the target object using a set of input 2D views.
2. Render the volume using volumetric ray marching.
3. Obtain a 2D segmentation (silhouette) of the target object from rendered raster images using a machine learning–based segmentation model.
4. Apply a noise removal method, called *stenciling*, to extract the target object from the scene and remove unnecessary voxels around the object.
5. Apply silhouette-based volume refinement.

The remainder of this paper is organized as follows. The related work section reviews prior approaches, including Neural Radiance Fields, and provides background on volumetric scene generation and the transformation of the scene into a custom, efficient volume data format. The methodology section describes our environment and details the proposed algorithms, including the silhouette-

*e-mail: gokseltokur@gmail.com
†e-mail: sxk144@shsu.edu
‡e-mail: doga@ou.edu

979-8-3195-0529-3/26/$31.00 ©2026 IEEE
DOI 10.1109/VRW70859.2026.00175

based volume refinement algorithm (S-VOLREF), the silhouette-based mesh refinement algorithm (S-MREF), and other techniques used in our pipeline. The results section presents experimental results that quantify reconstruction accuracy and demonstrate the effectiveness of the approach. Finally, the conclusion summarizes the work and outlines directions for future research.

2 Related Work

3D volume construction from sparse 2D views and the subsequent refinement of low-quality 3D models is a rapidly evolving research topic [25]. Multiple approaches have aimed to address these challenges. For instance, DeepVoxels [20] introduces a 3D embedding technique for neural network–based view synthesis, improving the quality of reconstructed 3D scenes. Occupancy Networks [15] focus on reducing memory usage while improving 3D model quality, and DeepSDF [18] excels at shape completion from point clouds. More recently, Neural Radiance Fields (NeRF) [16] and Instant Neural Graphics Primitives (Instant-ngp) [17] have emerged as high-performing techniques that represent complex 3D scenes using continuous volumetric functions, achieving state-of-the-art results in novel view synthesis.

NeRF [16] learns a 5D continuous function that maps 3D coordinates and viewing directions to radiance values, enabling the generation of novel views from sparse input images. By leveraging techniques such as structure-from-motion (SfM) to infer camera parameters and scene geometry, NeRF can synthesize missing viewpoints through volume rendering. While this yields high-fidelity view synthesis, extracting a clean, hole-free polygonal mesh from NeRF volumetric data remains challenging. Instant-ngp [17] improves upon NeRF by accelerating training and supporting more complex scenes, yet its output meshes often still suffer from noise and irregularities.

Several methods aim to directly extract high-quality surfaces from implicit representations. For example, NeuS [24] proposes a neural implicit surface reconstruction method that can produce smooth, detailed surfaces under controlled conditions. However, in our tests on a complex human model dataset, NeuS performed poorly, producing high average and Hausdorff distances from the ground truth. This suggests that while NeuS excels on certain benchmarks, it may struggle with real-world complexities and noise. Similarly, parametric human modeling approaches (e.g., SMPL [13]) rely on strong shape priors, which can limit their ability to capture fine details or handle non-ideal data conditions.

Our work builds upon these advances by introducing a silhouette-based refinement approach that integrates seamlessly with NeRF-generated volumetric data. Instead of relying solely on implicit representations or strong parametric priors, we leverage object silhouettes to guide noise removal, hole filling, and mesh refinement. In doing so, we address common shortcomings in NeRF-based reconstruction, such as surface gaps and noisy artifacts, and outperform methods like NeuS in challenging scenarios. Our silhouette-guided pipeline provides a robust alternative with improved accuracy and reliability for mesh extraction from volumetric data.

3 Methodology

3.1 Retrieving Volumetric Data and Volume Data Formatting

We employ Instant-ngp to construct the raw scene, which is subsequently retrieved into a custom file format. This format enables rendering with specific rendering algorithms, facilitating the use of our approach across different platforms.

Volumetric rendering and volumetric data exploration techniques are an active area of research [7]. As a result, volume rendering continues to evolve, driven by specialized frameworks and techniques developed for this purpose. Due to this rapid evolution and the diversity of volumetric data generation and storage methods, a universal volume data format standard remains elusive. This absence is largely due to variability in volumetric data properties such as physical characteristics, resolution, and dimensions, which depend on the application and acquisition technique. Therefore, after retrieving and saving the raw volume data from Instant-ngp, we developed a custom volume data format that restructures the raw data into a configuration better suited for rendering.

3.1.1 The Raw Volume Data Formatting

The raw volume data format includes the following components:

- Data Structure: A three-dimensional grid defined by dimensions $N(x)$, $N(y)$, and $N(z)$.
- Color Storage: Each grid point stores color information in four separate bytes representing red (R), green (G), blue (B), and alpha (A).
- Data Type: Each color component is stored as a byte within a float32 array. This is unusual because bytes do not require the full range of float32.
- Total Bytes per Element: 16 bytes (4 byte per color component).

3.1.2 The Custom Volume Data Formatting

Our custom volume data format includes the following components:

- Data Structure: A similar three-dimensional grid defined by $N(x)$, $N(y)$, and $N(z)$.
- Color Storage: RGBA values are packed into a single 4-byte structure (`uint32`) using bit manipulation.
- Data Type: `uint32`, enabling efficient packing and unpacking of multiple components into one compact unit.
- Total Bytes per Element: 4 bytes, storing all four components (RGB and A) together.

The advantages of our custom format are evident as it reduces file size compared to the raw format by packing RGBA values into 4 bytes rather than storing each component separately within a float32 array.

3.2 Volume Rendering

Volume rendering is widely used for visualizing volumetric data such as medical scans [1, 11]. In our pipeline, volume rendering serves two purposes: (i) producing realistic raster views of the NeRF volume for silhouette extraction, and (ii) supporting later stages such as noise removal, hole filling, and hollowing.

In general, volume data can be rendered either directly in voxel space (Direct Volume Rendering, DVR) or indirectly after extracting an explicit surface (Indirect Volume Rendering, IVR) [19, 26]. We employ both paradigms at different stages of our approach.

3.2.1 Direct Volume Rendering

Direct Volume Rendering (DVR) renders the volume without converting it to a polygonal surface. Rays are cast through the voxel grid and RGBA values are sampled and composited along the ray to determine the final pixel color [11]. In this work, we primarily use ray marching, where sampling occurs at fixed step intervals and accumulated color/opacity are updated progressively.

$$p_i = o + i\Delta s d, \quad \text{for } i = 0 \text{ to } (N-1) \tag{1}$$

where p_i is the current position at step i, o is the ray origin, d is the ray direction, N is the total number of steps, and Δs is the step size.

$$
\begin{aligned}
d &= C_i'.r \\
&= C_i'.g \\
&= C_i'.b \\
&= 10 \times C_i'.\alpha \qquad (2)\\
C_i.\text{rgb} &= C_i'.\alpha \times C_i'.\text{rgb} \\
&\quad +(1.0 - C_i'.\alpha) \times C_{i-1}.\text{rgb} \qquad (3)\\
C_i.\alpha &= C_i'.\alpha + (1.0 - C_i'.\alpha) \times C_{i-1}.\alpha \qquad (4)
\end{aligned}
$$

where C_i is the color at step i (a 4-dimensional rgba vector), and C_i' is the color contributed by the current position.

3.2.2 Indirect Volume Rendering

Indirect Volume Rendering (IVR) first extracts an explicit surface from the volume (e.g., using marching cubes) and then renders the resulting polygonal mesh with standard surface rendering [19, 26]. IVR is well-suited for mesh-based processing and enables efficient visualization once a clean surface is available.

In our pipeline, DVR is used to generate high-quality raster views for silhouette segmentation and refinement, while IVR is used after refinement to extract and render a polygonal mesh. Additionally, a first-hit voxel traversal is used to support hollowing by identifying surface voxels before mesh extraction.

3.2.3 Ray-Voxel Intersection Traversal Algorithm

Traversing a 3D volumetric grid can be computationally expensive, especially when the grid has high resolution or when traversal must be performed frequently [21].

In general, rendering volumetric data involves visiting every voxel (or a subset of voxels), depending on the algorithm and the goal. In our setting, we cast rays from each screen-space pixel of an orthographic camera. As the number of voxels increase, the computational cost grows accordingly. For example, if the grid has dimensions $N \times N \times N$, up to N voxels may be traversed along each dimension, yielding an $O(N^3)$ worst-case complexity. Consequently, more efficient traversal methods are needed.

Casting rays through a volumetric grid consists of three main steps:

1. Set up an orthographic camera, which allows rendering the volume without perspective distortion. In other words, the image scale remains constant regardless of the distance between the camera and the volume.
2. Cast rays from each screen-space pixel: iterate over all pixels on the screen, and for each one, generate a ray originating at the corresponding world-space position on the orthographic camera's near clipping plane, extending in the direction of the camera's view vector.
3. Perform ray marching along the rays and sample colors to determine each pixel. Ray marching is used to render the volume visually. As discussed earlier, ray marching is an iterative process in which a ray is advanced through the volume by a small step size determined by the desired sampling rate or voxel size. At each step along the ray, color sampling is performed. For our volume refinement approach, we employ a customized traversal method that examines the neighborhood of each voxel along the cast ray, using the Fast Voxel Traversal Algorithm (FVTA) [3].

The FVTA aims to minimize the number of floating-point operations while traversing a 3D voxel grid along a ray.

The Ray-Voxel Intersection Traversal Algorithm (RVI-T) has two main components. It is designed to traverse the volume efficiently and repeatedly during volume refinement, noise removal, and stenciling. We describe these stages later. The first component of RVI-T is the ray–box intersection algorithm. It determines whether a ray, originating from a 3D point with a given direction, intersects the volume. If an intersection occurs, it returns *true* along with the minimum and maximum parameter values at which the ray enters and exits the volume.

The ray–box intersection algorithm computes the minimum and maximum t values of the ray equation. We denote these as t_{min} and t_{max}, corresponding to where the ray enters and exits the volume, respectively. The following pseudocode summarizes the ray–box intersection procedure.

$$r(t) = o + t \cdot d \qquad (5)$$

where $r(t)$ is a point on the ray, o is the origin of the ray, t is the parameter along the ray, and d is the ray direction.

The second component, volume traversal, traces the path of the ray through the volume and returns the voxel path using the $tMin$ and $tMax$ values computed by the ray–voxel intersection method. If the ray–voxel intersection algorithm reports an intersection, the volume traversal algorithm computes the entry point using $tMin$. It then iteratively traverses the volume voxel by voxel along the ray by determining which voxel boundary is reached next at each step. The following pseudocode provides a high-level overview of the volume traversal algorithm.

After ray–volume intersection is completed, the next step is to extract human-body masks from different fields of view, as described in the following section.

3.2.4 Silhouette Segmentation for Masking

One of the main sources of noise is unnecessary or redundant voxels in NeRF-generated volumetric scenes is that NeRF reconstructs and renders the entire scene rather than explicitly isolating individual objects. Because NeRF models complex interactions among light, geometry, and materials, it can also capture irrelevant scene content (e.g., background structures, occlusions, and artifacts), which appears as noisy voxels surrounding the target object. Moreover, since NeRF learns a continuous function that maps 3D positions and viewing directions to radiance values, inaccuracies and approximations in the learned function can introduce noise and artifacts in the final volumetric representation. The rendering process, which integrates radiance along viewing rays, may further amplify these errors, particularly in regions with complex geometry or rapidly varying radiance. Consequently, noisy data and extraneous voxels in NeRF-generated volumes complicate accurate 3D mesh extraction and motivate robust refinement methods to improve volumetric quality and the resulting 3D models.

We use a trained human segmentation and masking machine-learning model [12] to segment the human body from 2D raster images rendered from the volumetric scene. Accurate segmentation is a crucial component of our approach because it enables both removal of scene noise and refinement of the target object using 2D images captured from multiple viewpoints. In addition, the segmentation model's ability to complete missing parts and correct defects in the 2D masks is important for achieving effective refinement.

Once the human-body masks are generated using the machine-learning model [12] as shown in Fig. 2, they are applied in the stenciling step to eliminate noise in the volumetric data.

3.3 Volume, Mesh Refinement, and Extraction of Polygonal Mesh

3.3.1 Stenciling for Noise Removal

As mentioned in the previous section, the first step of the pipeline is the removal of noise and unnecessary voxels from the volumetric NeRF scene. We refer to this stage as *stenciling*. After generating and accurately segmenting a 2D raster image of the volume from a

Algorithm 1 Ray-Voxel Intersection Algorithm

Require: $o, d, VolumeMin, VolumeMax$
// Calculate inverse direction
$inverseDirection.x = 1/d.x$
$inverseDirection.y = 1/d.y$
$inverseDirection.z = 1/d.z$
if $inverseDirection.x \geq 0$ **then**
  $tMin = (VolumeMin.x - o.x) \times inverseDirection.x$
  $tMax = (VolumeMax.x - o.x) \times inverseDirection.x$
else
  $tMin = (VolumeMax.x - o.x) \times inverseDirection.x$
  $tMax = (VolumeMin.x - o.x) \times inverseDirection.x$
end if
if $inverseDirection.y \geq 0$ **then**
  $tMinY = (VolumeMin.y - o.y) \times inverseDirection.y$
  $tMaxY = (VolumeMax.y - o.y) \times inverseDirection.y$
else
  $tMinY = (VolumeMax.y - o.y) \times inverseDirection.y$
  $tMaxY = (VolumeMin.y - o.y) \times inverseDirection.y$
end if
// No intersection, return false.
if $(tMin > tMaxY)$ **or** $(tMinY > tMax)$ **then**

  return $(false, -1, -1)$
end if
if $tMinY > tMin$ **then**
  $tMin = tMinY$
end if
if $tMaxY < tMax$ **then**
  $tMax = tMaxY$
end if
if $inverseDirection.z \geq 0$ **then**
  $tMinZ = (VolumeMin.z - o.z) \times inverseDirection.z$
  $tMaxZ = (VolumeMax.z - o.z) \times inverseDirection.z$
else
  $tMinZ = (VolumeMax.z - o.z) \times inverseDirection.z$
  $tMaxZ = (VolumeMin.z - o.z) \times inverseDirection.z$
end if
// No intersection, return false.
if $(tMin > tMaxZ)$ **or** $(tMinZ > tMax)$ **then**

  return $(false, -1, -1)$
end if
if $tMinZ > tMin$ **then**
  $tMin = tMinZ$
end if
if $tMaxZ < tMax$ **then**
  $tMax = tMaxZ$
end if
// Intersection, return true, $tMin$ and $tMax$.
return $(true, tMin, tMax)$

Algorithm 2 Volume Traversal Algorithm

Require: $o, d, volumeSize$
Initialize $path$ as an empty list.
Calculate $(intersection, tmin, tmax)$ using the RayBoxIntersection algorithm.
if not $intersection$ **then**
  return null
else
  if $tmin < 0$ **then**
    $tmin = 0$
  end if
  Compute current voxel positions $tVoxel_{x,y,z}$ and direction of steps $step_{x,y,z}$ as 1 if the ray's direction is positive, otherwise -1.
  $voxelMax_{x,y,z} = tVoxel_{x,y,z} \times volumeSize$
  $tMax_{x,y,z} = tmin + \frac{(voxelMax_{x,y,z} - o_{x,y,z})}{d_{x,y,z}}$
  $tDelta_{x,y,z} = \frac{1.0}{|d_{x,y,z}|}$
  while $x \leq volumeSize$ **and** $x \geq 1$ **and** $y \leq volumeSize$ **and** $y \geq 1$ **and** $z \leq volumeSize$ **and** $z \geq 1$ **do**
    Add the current voxel coordinates (x, y, z) to the $path$.
    if $tMax_x < tMax_y$**and**$tMax_x < tMax_z$ **then**
      $x += step_x$
      $tMax_x += tDelta_x$
    else if $tMax_y < tMax_z$ **then**
      $y += step_y$
      $tMax_y += tDelta_y$
    else
      $z += step_z$
      $tMax_z += tDelta_z$
    end if
  end while
end if
return $path$ // Return traversed path.

given viewpoint, the corresponding target-object mask is used as a filter on the NeRF-generated volume data. These silhouettes (stencils), captured from multiple angles, serve as guides that determine which screen points should be used to cast rays through the volume using the Ray–Voxel Intersection Traversal algorithm. This process discards irrelevant voxels and isolates the target object within the volume.

The 3D stenciling procedure operates as follows. For each pixel that is empty in the mask image (see Fig. 3, black regions) but filled in the base raster image, a ray is cast into the volume. These rays traverse the volume and clear all voxels along their paths. The camera then orbits around the volume, repeating this noise-clearance process at predefined angular increments until a full rotation is completed, as shown in Fig. 4. The results before and after stenciling are shown in Fig. 5.

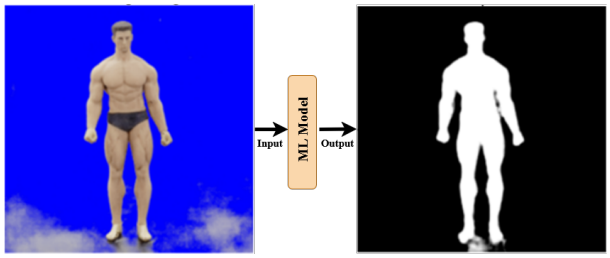

Figure 2: 2D masking of a target object inside the rendered volume.

3.3.2 Silhouette-Based Volume Refinement Algorithm (S-VOLREF)

In the noise-clearance step, the method called *stenciling* leverages the target-object mask to eliminate noise and unnecessary voxels in the NeRF-generated volumetric scene. This process relies on accurate segmentation of the target object. However, noise removal alone is insufficient for extracting a high-quality mesh or achieving photorealistic volumetric rendering, because NeRF's approximation can introduce inaccuracies, defects, and holes in the reconstructed object.

To address these issues, we introduce a silhouette-based volume refinement technique that follows a similar workflow to stenciling. This refinement uses masks of the 3D volume object derived from 2D images captured from different angles, produced by a trained machine-learning segmentation model. The volume refinement step differs from stenciling in how the base raster image and the mask image are compared. As in stenciling, we perform a pixel-by-pixel comparison between the rendered volume's base raster image and the mask image. However, the goal here is to identify pixels that

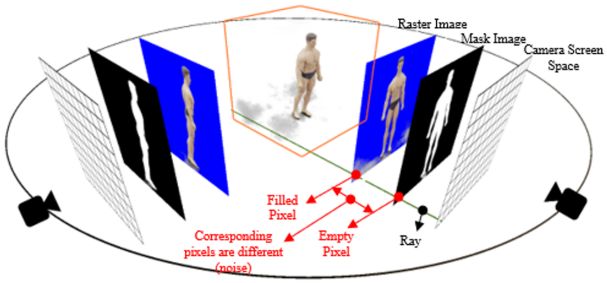

Figure 3: Visualization of stenciling process. Green line represents a ray.

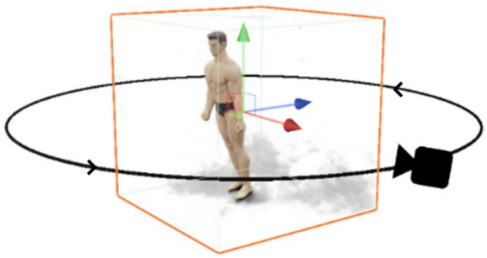

Figure 4: Camera orbital rotation around the volume.

should be filled but are currently empty in the volume, rather than pixels that should be cleared. By doing so, we improve the volumetric representation of the target object by filling holes and refining defects. This improvement enables more accurate mesh extraction from the refined volume using the Marching Cubes algorithm.

3.3.3 Filling Holes and Refining Defects

Filling holes and refining defects in the volumetric rendering is a key part of the volume refinement process. Such imperfections may arise from errors in the initial 2D image acquisition or from limitations of the NeRF reconstruction. The objective of this stage is to improve the quality and accuracy of the volumetric representation, thereby enabling extraction of a higher-quality 3D mesh using an isosurface extraction method, specifically the Marching Cubes algorithm.

In this stage, the 2D masks of the 3D volume object are used again. Unlike the stenciling (noise-reduction) stage, which removes voxels that do not belong to the target object (including noisy voxels), the volume refinement algorithm focuses on identifying and filling holes and missing parts within the target object's volumetric data. To achieve this, a pixel-by-pixel comparison is performed between the base image of the rendered volume and the corresponding mask image. The algorithm compares the current state of the volume (represented by the base image) with the expected state (represented by the mask). If a pixel is empty in the base image but should be filled according to the mask, the algorithm casts a voxel-filling ray from the corresponding screen-space pixel through the volume. Fig. 6 and Fig. 7 illustrate this process. The resulting improvement in the quality of the extracted mesh is illustrated in Fig. 8, which compares the meshes reconstructed from the raw and refined volumes.

3.3.4 Intensity and Color Sampling

After identifying suitable coordinates for filling, the algorithm computes the intensity of each filled voxel, taking into account the characteristics of volumetric data and rendering. Various methods can be used to assign voxel intensity. One simple approach is to use a predetermined constant value for all filled voxels. However, our goal is to compute an intensity for each filled voxel that closely matches its neighborhood by blending the intensities and colors of neighboring voxels.

Each filled voxel should integrate smoothly into its surroundings, filling holes and refining the volume while preserving the photorealistic quality of the rendering. Upon completion of the refinement process, the modified regions should blend seamlessly with the rest of the volume. Additionally, when correcting defects in the 3D volumetric representation, it is crucial to preserve the original structure of the volume. Therefore, we employ an appropriate stopping condition to prevent rays from overfilling voxels and potentially altering the object's structure. This stopping criterion helps preserve the object while refining its representation using neighborhood-informed intensity values.

3.3.5 Overfilling Voxel Elimination

To prevent overfilling and prioritize hole filling and defect refinement, we use intensity values together with a threshold. These values, coupled with a threshold, ensure that the volume is filled appropriately without excessive growth. The neighbor-blending method adjusts intensity values gradually outward from the filled voxel, which helps prevent overfilling. In addition, we employ a neighbor-count threshold to avoid unnecessary filling, ensuring that only voxels with a sufficient number of neighboring voxels are filled during the refinement process.

3.3.6 Hollowing Stage

After refining the volume and before applying mesh extraction techniques, the hollowing stage follows. This stage creates a hollow version of the refined object, simplifying the volume and preparing it for efficient mesh extraction using the Marching Cubes algorithm [14]. Hollowing removes internal voxels that do not belong to the external surface of the object, retaining only voxels that define the outer boundary.

Hollower rays are cast from the pixel coordinates of the mask used in previous steps through the volume, covering a full 360-degree view across two axes. Each ray marks the first encountered voxels as surface voxels. After all rays have been cast, the algorithm removes unmarked (interior) voxels and preserves the marked voxels that define the outer surface of the object. At the end of the hollowing stage, a simplified volume representation remains, ready for the isosurface extraction step, with voxels primarily defining the external surface of the object.

Following the volume refinement process, the algorithm produces refined volumetric data of the target object for subsequent processing. The refined volume is then passed to the Marching Cubes algorithm to extract isosurfaces and generate the polygonal mesh of the target model.

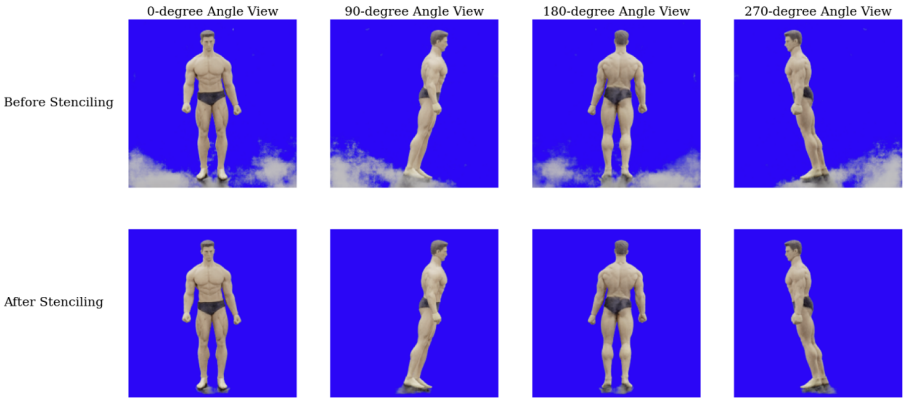

Figure 5: Stenciling results.

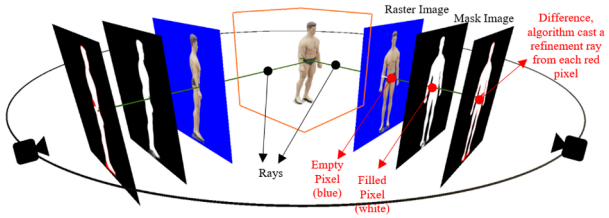

Figure 6: Visualization of silhouette-based volume refinement process. Green lines represent refinement rays, which are cast from the screen space coordinates of different pixels (which are black pixels in the mask, blue pixels in the raster image, and red pixels in the difference image) through the volume.

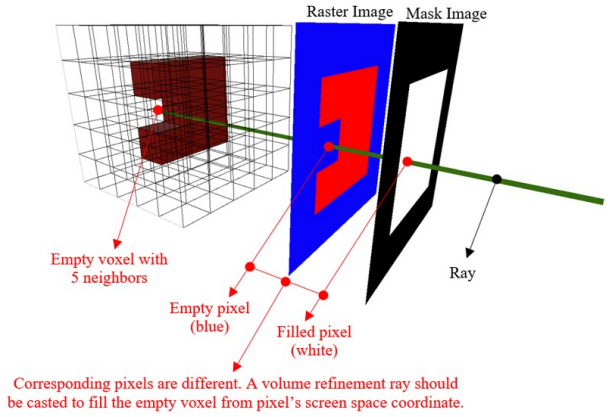

Figure 7: Visualization of volume refinement ray.

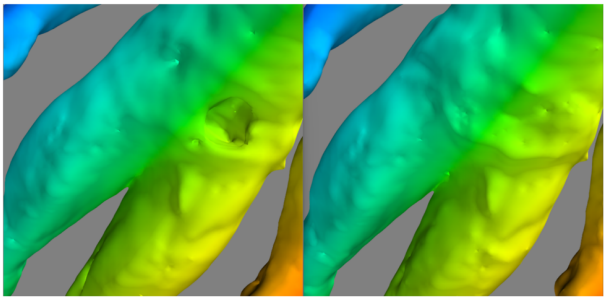

Figure 8: Reconstructed mesh from raw volume without applying silhouette-based volume refinement on the left, reconstructed mesh from refined volume using silhouette-based volume refinement algorithm (S-VOLREF).

3.3.7 Mesh Extraction

Extracting a polygonal mesh from a 3D discrete scalar field, such as 3D volume grid data, is a challenging task. The Marching Cubes algorithm was developed to address this challenge and has become the de facto standard isosurface extraction algorithm in scientific visualization. It is widely employed in various fields, including medical imaging (e.g., magnetic resonance imaging (MRI), computed tomography (CT), and single-photon emission computed tomography (SPECT)), where it constructs triangle models from 3D volume data. These modalities produce volumetric data similar to that used in this study. In our pipeline, Marching Cubes is used to extract a polygonal mesh from the refined volume produced in earlier steps.

However, the raw mesh produced by Marching Cubes typically appears blocky and rough rather than smooth and continuous because it approximates the surface using discrete cubes (voxels). Consequently, additional processing steps, including smoothing algorithms, are required to refine the output mesh.

3.3.8 Mesh Smoothing

After the refined volume data is processed by the Marching Cubes algorithm, it produces a polygonal mesh representing the 3D object. As mentioned earlier, the raw mesh often appears jagged due to discretization in the Marching Cubes process.

To reduce roughness caused by curvature and improve the surface quality of the raw mesh, we employ Laplacian smoothing [22],

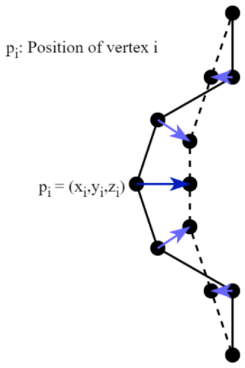

Figure 9: 2D representation of Laplacian smoothing.

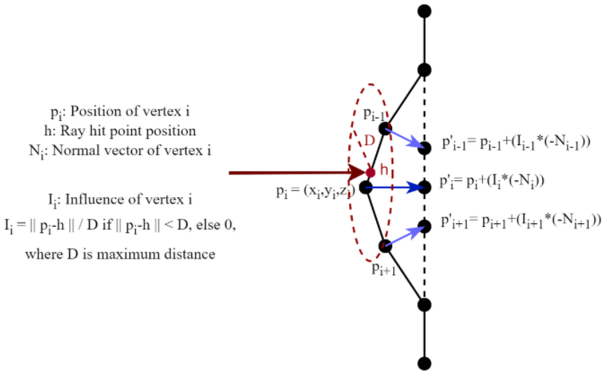

Figure 10: Formulation and 2D representation of mesh refinement algorithm.

a widely used technique in mesh processing. This method aims to minimize the Laplacian error, which measures local curvature at each vertex. Specifically, the Laplacian value is computed as the distance between a vertex position and the average position of its neighboring vertices. Laplacian smoothing is applied to each vertex as follows:

$$p'_i = \frac{1}{N} \sum_{j=1}^{N} p_j \tag{6}$$

where p'_i is the new position of vertex i, p_j is the position of the j-th neighbor of vertex i, and N is the number of neighboring vertices of vertex i.

An exemplary visualization of Laplacian smoothing is illustrated in Fig. 9.

The algorithm iterates over the mesh vertices with the objective of minimizing the Laplacian value, typically controlled by a predefined number of iterations or a stopping threshold. However, there is a trade-off between visual smoothness and the preservation of surface details. Therefore, selecting an appropriate stopping condition is crucial.

3.3.9 Silhouette-Based 3D Mesh Refinement Algorithm (S-MREF)

As an additional stage in the pipeline after mesh generation and smoothing, we further refine the mesh by adjusting its vertices. This step can be viewed as a smoothing operation, but it differs from Laplacian smoothing and other traditional techniques. Here, we adapt the silhouette-based volume refinement approach used earlier in the pipeline to operate on polygonal meshes. The goal is to refine and smooth the surface, particularly by reducing irregularities in specific regions of the mesh.

The process casts rays from the viewpoint of the 2D mask onto the 3D mesh, targeting vertices that contribute to surface irregularities. Specifically, when discrepancies are detected between the base 2D projection of the mesh and the 2D mask from that viewpoint, the algorithm casts a ray from the corresponding pixel coordinate through the mesh. The vertex closest to the intersection point of the ray then deviates the most from the expected silhouette.

Once such vertices are identified, the algorithm displaces them inward along their normal vectors. To avoid introducing new sharp transitions on the surface, we compute a distance-based influence value for the surrounding vertices using a simple linear falloff function. The displacement is then applied to these neighboring vertices proportionally to their influence values. Fig. 10 illustrates the underlying equations and the resulting displacements of surrounding vertices.

Table 1: Specifications of 3D printer.

	Printer 1
Build Volume	165 mm x 165 mm x 300 mm
Layer Thickness	110 microns, 0.004 in
Technology	Selective Laser Sintering

By adjusting the positions of vertices that contribute to bumpy regions using the 2D silhouette, surface roughness and irregularities can be reduced, thereby improving overall mesh accuracy. The outcome of the proposed pipeline is presented in the following section.

4 Results and Discussion

The ground truth mesh is a polygonal mesh of a 3D virtual model. We then fabricated physical copies using a 3D printer. Ensuring printer's accuracy and the precision was crucial to minimize printing-induced errors and enable reliable evaluation. Therefore, we selected a 3D printer with high accuracy and precision specifications. The models were printed at three different scales: downscaled, original scale, and upscaled relative to the original mesh. Fig. 11 illustrates the 3D printed models at these scales. Detailed specifications of the 3D printer are provided in Tab. 1.

After printing the ground truth object and obtaining physical copies, we scanned them using two different laser scanners to assess the accuracy of the reconstructed meshes and compare them against the mesh produced by our pipeline. The make, model, and specifications of these scanners are listed in Tab. 2. To generate the NeRF dataset for our pipeline, we used two different cameras with distinct specifications and resolutions; the camera specifications are given in Tab. 3.

4.1 Evaluation Metrics

We utilized three primary metrics to compare meshes generated by our volume refinement and mesh reconstruction pipeline with the ground truth mesh (the 3D virtual model) and meshes scanned using physical scanners. These comparisons quantify the accuracy and reliability of our volume and mesh refinement pipeline. The evaluation metrics include Average Point Cloud Distance (A-PCD) and Hausdorff Distance (HD). Additionally, we implemented a shader-based metric to quantify the percentage of the mesh surface affected by holes and defects. To ensure accurate alignment with the ground truth, we used the Iterative Closest Point (ICP) algorithm [4] during the A-PCD and HD evaluations.

A-PCD measures the average distance between corresponding points in a given mesh and the ground truth mesh. After con-

Table 2: Specifications of scanners.

	Low-cost Scanner	High-cost Scanner
Scan Accuracy	0.43 mm	0.04 mm
Maximum Scan Size	180 mm x 250 mm	240 mm x 310 mm
Optics	HD CMOS sensor, and 2 lasers	2 monochrome cameras with LEDs

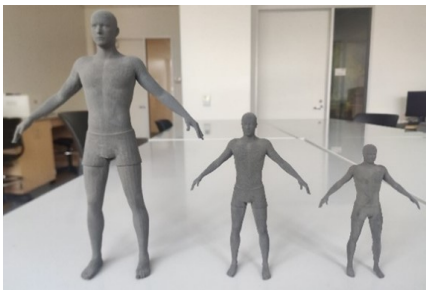

Figure 11: 3D printed models of various sizes.

verting the input meshes to point clouds, we sample a predefined number of points from each point cloud. A smaller A-PCD indicates higher similarity between the ground truth mesh used for 3D printing and the meshes generated by our pipeline or obtained from scanner measurements of the printed models.

However, A-PCD is sensitive to variations in point sampling and distribution. Therefore, we also report the Hausdorff Distance (HD), which captures the maximum deviation between two point clouds. Specifically, HD represents the greatest distance from a point in one sample to the closest point in the other sample among the sampled points. The formulas for A-PCD and HD are presented in Equation 7–9.

Let,

$$P = \{p_1, p_2, p_3, \ldots, p_n\} \quad \text{and} \quad P' = \{p'_1, p'_2, p'_3, \ldots, p'_n\} \tag{7}$$

be two point clouds with n sample points in 3D space. Then, A-PCD is defined in Equation 8, and HD is defined in Equation 9.

$$A_{PCD}\left(P, P'\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\min_{j=1}^{n}\left(\|p_i - p'_j\|\right)\right) \tag{8}$$

$$HD\left(P, P'\right) = \max\left\{\begin{array}{l}\max\limits_{i=1}^{n}\left(\min\limits_{j=1}^{n}\left(\|p_i - p'_j\|\right)\right), \\ \max\limits_{j=1}^{n}\left(\min\limits_{i=1}^{n}\left(\|p_i - p'_j\|\right)\right)\end{array}\right\} \tag{9}$$

The mesh reconstructed from the NeRF volume using marching cubes, without applying our volume refinement, often exhibits defects and holes. Therefore, a key objective of our volume refinement algorithm is to mitigate such surface artifacts. To quantify the effectiveness of our method in preventing surface defects, we introduce a metric named Shader-Based Hole Detection on Mesh Surface (S-HDMS). This metric leverages back-face culling [2] to evaluate holes, defects, and overall surface integrity.

S-HDMS distinguishes front and back faces of a 3D mesh by coloring vertices based on the dot product between the surface normal and the camera view direction. After rendering the mesh, we apply the Flood Fill algorithm [8] to separate the outer background from the rendered model region across multiple viewpoints. We then compute the percentage of hole pixels, which correspond to pixels that reveal the background or back-facing regions.

In Equation 10, $C(v)$ represents the color assigned to vertex v based on $\vec{n} \cdot \vec{v}$, which reflects the angle between the surface normal and the view direction. After flood filling the background, we count the pixels corresponding to holes as shown in Equation 11.

$$C(v) = \text{color}(\vec{n} \cdot \vec{v}) \tag{10}$$

$$\text{Percent}_{\text{Holes}} = \frac{\text{Number of hole pixels}}{\text{Total number of pixels in the model region}} \tag{11}$$

The results for these metrics are presented in the following section.

4.2 Results

The A-PCD and HD evaluation results are presented in Tabs. 4 to 6. Tab. 4 compares the meshes of downscaled 3D printed models scanned with 3D scanners, the raw mesh directly constructed from the NeRF volume, and the mesh refined by our volume refinement algorithm. Both the raw mesh and our refined mesh use volumes with identical bounding-box sizes. Tab. 5 presents the results for the original-scale model, while Tab. 6 shows the results for the upscaled model.

The high error rates in the raw NeRF mesh primarily stem from volumetric noise, which our algorithm addresses through silhouette-based noise removal techniques. In comparison, physical scanners exhibit poorer performance when scanning upscaled models due to fixed scanning environments and limited coverage. Our pipeline, by contrast, is scale-free, enabling it to handle large-scale objects without such restrictions. Tab. 7 illustrates the effectiveness of our mesh refinement algorithm, demonstrating a reduction in A-PCD and HD values toward the ground truth when applying our S-VOLREF method.

Additionally, Tab. 8 presents the percentages of holes and defects in reconstructed meshes based on S-HDMS. High-quality scanners and some reconstruction workflows often leave noticeable holes due to coverage limitations or data noise. Our volume refinement algorithm effectively addresses these issues. Fig. 12 showcases the resulting meshes, highlighting that our silhouette-guided pipeline delivers clean, hole-free surfaces even under challenging conditions.

As a further comparison, we evaluated NeuS [24], a state-of-the-art neural surface reconstruction technique, on our dataset. Using default parameters provided in the NeuS codebase, we trained NeuS for approximately eight hours on an NVIDIA RTX 4070 Ti Super. Despite the extensive training, NeuS yielded poor reconstruction quality on our complex human dataset (see Fig. 12), resulting in high average point-to-point and Hausdorff distances. A-PCD and HD comparisons between the NeuS reconstruction and our mesh are reported in Tab. 7. This contrast underscores the robustness of our approach. While NeuS and similar methods may excel on controlled or synthetic benchmarks, they can struggle with real-world complexities. In scenarios where these advanced methods fail, our silhouette-guided pipeline demonstrates its effectiveness and reliability in producing accurate, hole-free meshes.

5 Conclusion

The silhouette-based volume refinement algorithm introduced in this research, together with a 3D mesh reconstruction pipeline that applies the state-of-the-art neural rendering method Neural Radiance Fields (NeRF) to volumetric data, addresses the challenges of reconstructing 3D meshes from 2D images. The pipeline integrates key techniques such as silhouette-based noise removal, extraction of object-specific volumetric renderings, and refinement of volumetric representations. Together, these techniques enable the accurate reconstruction of 3D meshes from 2D image data.

Our pipeline achieved significant improvements in the quality of reconstructed 3D meshes. Through quantitative comparisons,

Table 3: Specifications of cameras.

	Camera 1	Camera 2
Modules	12 MP, f/1.8, 28mm (wide), PDAF, OIS, 12 MP, f/2.8, 57mm (telephoto), PDAF, 2x optical zoom	64 MP, f/1.8, 26mm (wide), 1/1.7", 0.8µm, PDAF, OIS, 12 MP, f/2.2, 123 degree (ultrawide), 1.12µm, 5 MP, f/2.4, (macro), 5 MP, f/2.4, (depth)
Video Frame Size	720x1280 pixels	1080x1920 pixels
Video Frame Rate	30 frames/second	30 frames/second

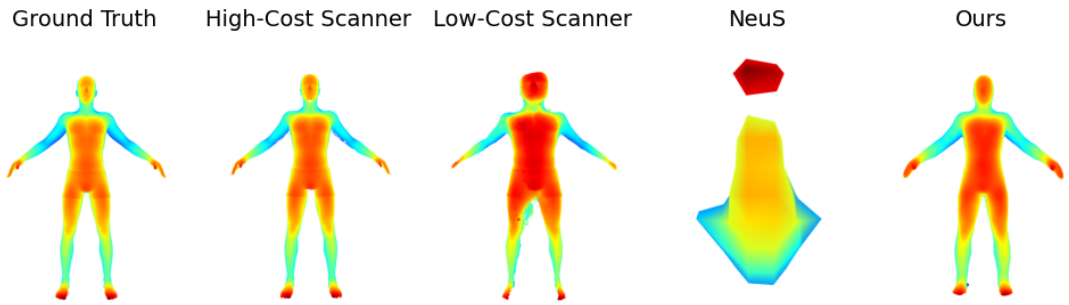

Figure 12: Mesh results.

Table 4: A-PCD and HD metrics of meshes reconstructed from downscaled 3D printed models.

Scanners/Scanning Techniques	A-PCD	HD
Low-Cost Scanner	1.89	9.28
High-Cost Scanner	0.48	5.12
Raw NeRF + Marching Cubes, Camera 1	31.75	126.6
Raw NeRF + Marching Cubes, Camera 2	27.36	137.97
S-VOLREF, Camera 1	1.02	8.55
S-VOLREF, Camera 2	0.86	13.61

Table 5: A-PCD and HD metrics of meshes reconstructed from original-scale 3D printed models.

Scanners/Scanning Techniques	A-PCD	HD
Low-Cost Scanner	2.28	10.64
High-Cost Scanner	0.53	6.33
Raw NeRF + Marching Cubes, Camera 1	38.17	154.32
Raw NeRF + Marching Cubes, Camera 2	11.96	119.77
S-VOLREF, Camera 1	1.00	11.05
S-VOLREF, Camera 2	0.84	9.99

we validated the effectiveness of our volume and mesh refinement algorithms using metrics such as Average Point Cloud Distance, Hausdorff Distance, and a shader-based hole-percentage measure on the mesh surface. These results demonstrate that our approach enables precise mesh extraction by eliminating noise and refining volumetric representations.

The improved results obtained with our approach also indicate room for further improvement. In future work, we aim to integrate more advanced machine learning models that provide higher precision in object masking and better completion of missing parts in 2D images. Such models would further improve the efficiency of our noise removal, S-VOLREF, and S-MREF stages.

In summary, Silhouette Sculptor advances automated and accurate 3D mesh reconstruction from 2D images. The proposed pipeline and algorithms provide a foundation for further improvements and have potential applications in medical imaging, video games, and industrial manufacturing.

Table 6: A-PCD and HD metrics of meshes reconstructed from upscaled 3D printed models.

Scanners/Scanning Techniques	A-PCD	HD
Low-Cost Scanner	4.97	44.14
High-Cost Scanner	6.74	95.44
Raw NeRF + Marching Cubes, Camera 1	47.54	217.84
Raw NeRF + Marching Cubes, Camera 2	42.18	210.64
S-VOLREF, Camera 1	2.09	36.92
S-VOLREF, Camera 2	1.86	16.82

Table 7: A-PCD and HD comparison of raw mesh, NeuS reconstructed mesh, volume refined mesh, and both volume and mesh refined mesh (Camera 2).

Method/Technique	A-PCD	HD
Raw NeRF + Marching Cubes, Camera 2	11.96	119.80
NeuS, Camera 2	18.36	104.86
S-VOLREF, Camera 2	0.84	10.06
S-VOLREF + S-MREF, Camera 2	0.80	9.98

Table 8: Hole percentage (%) of reconstructed meshes based on S-HDMS for different scanners/techniques and model scales.

Scanner/Technique	Downscaled Model	Original Scale Model	Upscaled Model
Low-Cost Scanner	0.20	0.11	0.09
High-Cost Scanner	1.49	1.48	4.44
Raw	8.18	10.41	14.94
Our Algorithms	0.07	0.04	0.02

ACKNOWLEDGMENTS

This project is partially supported by the National Institutes of Health (NIH) award NIAMS R44AR075481.

REFERENCES

[1] The national library of medicine's visible human project. U.S. National Library of Medicine. Accessed July 19, 2023. 2

[2] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering*. A K Peters, Ltd., third ed., 2008. doi: 10.1201/b10644 8

[3] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. In *Eurographics '87*, 1987. 3

[4] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987. doi: 10.1109/TPAMI.1987.4767965 7

[5] E. Cerezo, F. Pérez, X. Pueyo, F. J. Seron, and F. X. Sillion. A survey on participating media rendering techniques. *The Visual Computer*, 21(5):303–328, 2005. doi: 10.1007/s00371-005-0287-1 1

[6] F. Dinc, K. Oumimoun, W. Kwabla, S. Kockara, T. Halic, S. Arikatla, and S. Ahmadi. Towards real-time bone drilling simulation for anchor placement in vr based arthroscopic rotator cuff surgery simulation. In *AMIA Joint Summits on Translational Science Proceedings*, pp. 178–185, 2022. 1

[7] W. Dong, C. Choy, C. Loop, O. Litany, Y. Zhu, and A. Anandkumar. Fast monocular scene reconstruction with global-sparse local-dense grids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[8] S. Edelkamp. Shortest paths. In *Algorithmic Intelligence: Towards an Algorithmic Foundation for Artificial Intelligence*, pp. 35–48. Springer, Cham, 2023. 8

[9] J. J. Koenderink and A. J. van Doorn. Affine structure from motion. *Journal of the Optical Society of America A*, 8(2):377, 1991. doi: 10.1364/JOSAA.8.000377 1

[10] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994. doi: 10.1109/34.273735 1

[11] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988. doi: 10.1109/38.511 2

[12] S. Lin, L. Yang, I. Saleemi, and S. Sengupta. Robust high-resolution video matting with temporal guidance. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022. doi: 10.1109/WACV51458.2022.00319 3

[13] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics*, 34(6):248:1–248:16, 2015. 2

[14] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1987. doi: 10.1145/37401.37422 5

[15] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. doi: 10.1109/CVPR.2019.00459 2

[16] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Computer Vision – ECCV 2020*, pp. 405–421, 2020. doi: 10.1007/978-3-030-58452-8_24 1, 2

[17] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, 2022. doi: 10.1145/3528223.3530127 2

[18] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. doi: 10.1109/CVPR.2019.00025 2

[19] B. Preim and C. Botha. Virtual endoscopy. In *Visual Computing for Medicine: Theory, Algorithms, and Applications*, pp. 509–536. Morgan Kaufmann, Waltham, MA, 2014. 2, 3

[20] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. doi: 10.1109/CVPR.2019.00254 2

[21] B. Smits. Efficiency issues for ray tracing. In *ACM SIGGRAPH 2005 Courses*, 2005. doi: 10.1145/1198555.1198745 3

[22] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 2004. doi: 10.1145/1057432.1057456 6

[23] Y. Su. The application of 3d technology in video games. *Journal of Physics: Conference Series*, 1087:062024, 2018. doi: 10.1088/1742-6596/1087/6/062024 1

[24] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 27171–27183, 2021. 2, 8

[25] Y. Wu, Z. Zou, and Z. Shi. Remote sensing novel view synthesis with implicit multiplane representations. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–13, 2022. doi: 10.1109/TGRS.2022.3197409 2

[26] L.-Q. Yan. Realistic rendering in 'details.'. *IEEE Computer Graphics and Applications*, 41(4):20–26, 2021. doi: 10.1109/MCG.2021.3077918 2, 3