VortSDF: 3D Modeling with Centroidal Voronoi Tesselation on Signed Distance Field
Supplemental Material

1. Overview

In this supplementary material we provide:

- Details on the ray marching algorithm in a tetrahedral mesh.
- An additional evaluation of the different methods with the PSNR, which reflects quality of the appearance model generated by each method.
- Qualitative visualizations of the results in the associated video.

2. Ray marching in the tetrahedral mesh

Computing the intersections of a ray, from a camera center through a pixel, with a tetrahedral mesh consists first in finding the entry point and then iteratively identifying the next neighboring tetrahedron to visit in order to find the next intersection point. Finding the entry point can be a bottleneck if all the outside faces of the tetrahedral mesh needed to be tested. Instead, in our proposal we take advantage of the tetrahedral data structure by incorporating the camera centers into the triangulation. Identifying the entry tetrahedron for a given ray simply consists in testing the tetrahedra that contain the ray camera center. The algorithm for ray marching is reported in pseudo code in the algorithms 1-4.

Algorithm 1 Ray marching algorithm

```
INPUT A 3D ray $(o, ray_v)$ with direction $ray_v$ and start point the camera $id_o$ at location $o$.

A. Build projection base for the ray (Alg. 2).

B. Identify entry tetrahedon (Alg. 3).

C. Walk though the tetrahedral mesh (Alg. 4).

OUTPUT A set of segments that intersect the tetrahedral mesh.
```

Algorithm 2 A. Build projection base for the ray

```
INPUT A 3D ray $(o, ray_v)$ with direction $ray_v$ and start point the camera $id_o$ at location $o$.

$k_{min} \leftarrow argmin_i(|ray_v[i]|)$
$k_{max} \leftarrow argmax_i(|ray_v[i]|)$
$\mathbf{u}[k_{min}] \leftarrow 0$
$\mathbf{u}[(k_{min}+1)\%3] \leftarrow \frac{ray_v[(k_{min}+2)\%3]}{ray_v[k_{max}]}$
$\mathbf{u}[(k_{min}+2)\%3] \leftarrow \frac{-ray_v[(k_{min}+1)\%3]}{ray_v[k_{max}]}$
$\mathbf{t} \leftarrow ray_v \wedge \mathbf{u}$
$k_{min} \leftarrow argmin_i(|t[i]|)$
$k_{max} \leftarrow argmax_i(|t[i]|)$
$\mathbf{v}[0] \leftarrow \frac{t[0]}{t[3-k_{min}-k_{max}]}$
$\mathbf{v}[1] \leftarrow \frac{t[1]}{t[3-k_{min}-k_{max}]}$
$\mathbf{v}[2] \leftarrow \frac{t[2]}{t[3-k_{min}-k_{max}]}$

OUTPUT Projection vectors $\mathbf{u}$ and $\mathbf{v}$.
```

2.1. Algorithm A

Algorithm 2 builds a local coordinate system centered at the camera location and oriented in the direction of the normal vector. Vectors $\mathbf{u}$ and $\mathbf{v}$ in alg. 2 are the two orthogonal vectors orthogonal to the ray direction ray_v. They are used to project the summits of the tetrahedra onto the 2D plane perpendicular to the ray and centered on the camera center.

2.2. Algorithm B

Algorithm 3 Identifies the first tetrahedron that is intersected by the ray by testing all tetrahedra that contain the camera center as one of its summits are tested. Each time we test if the face opposite to the camera center (which is the start point) intersects the ray. If such a case is found then the tetrahedron is the first tetrahedron to start ray marching.

2.3. Algorithm C

Algorithm 4 walks through the tetrahedral mesh. At each iteration, the summits of the current tetrahedron are pro-

Algorithm 3 B. Identify entry tetrahedon

```
INPUT A 3D ray $(o, ray_v)$ with direction $ray_v$ and start point the camera $id_o$ at location $o$. Projection vectors $\mathbf{u}$ and $\mathbf{v}$.

for $tet$ in $adj[id_o]$ do
  ▷ $adj[id_o]$ is the list of tetrahedra that contain vertex $o$.
  $id[0..3] \leftarrow$ four summits ids of $tet$
  Organise summits ids so that $id[3] == id_o$
  for $j$ in $[0..2]$ do
    $\mathbf{v_{curr}} \leftarrow \mathbf{s[id[j]] - o}$
    ▷ $\mathbf{s}$ is the list of 3D vertices of the tetrahedral mesh
    $p[j] \leftarrow [\mathbf{u} \cdot \mathbf{v_{curr}}, \mathbf{v} \cdot \mathbf{v_{curr}}]$
  end for
  if Origin in $(p[0], p[1], p[2])$ then
    $T_{curr} \leftarrow tet$
    break
  end if
end for

OUTPUT The id of the first tetrahedron intersected by the ray.
```

jected into the 2D plane constructed in algorithm 2 and the exit face is identified as the one that contains the origin in the 2D projected plane. Note that the entry face is not considered. Once the next tetrahedron is identified indices of the summits are re-organized using the XOR operator to ensure that always the firs three summits correspond to the entry face and the fourth summit is the summit opposite to the entry face. This greatly simplifies computations.

Algorithm 4 C. Walk though the tetrahedral mesh

```
INPUT A 3D ray $(o, ray_v)$ with direction $ray_v$ and start point the camera $id_o$ at location $o$. Projection vectors $\mathbf{u}$ and $\mathbf{v}$. Current tetrahedron $T_{curr}$ with summits ids $TetID$.

INPUT $id[0]$, $id[1]$ and $id[2]$ are indices of entry face in the previously visited tetrahedron.

INPUT $id_e$ is id of exit face in previously visited tetrahedron (i.e. opposite summit $id[id_e]$ is not in current tetrahedron).

1. $id[id_e] \leftarrow id[3]$
  ▷ Now $id[0]$, $id[1]$ and $id[2]$ are indices of entry face in the current tetrahedron.
2. $id[3] \leftarrow id[0] \oplus id[1] \oplus id[2] \oplus TetID[3]$
                                        ▷ $\oplus$ is the XOR operator
3. $v_n \leftarrow v[id[3]] - o$
4. $p[3] \leftarrow (v_n \cdot u, v_n \cdot v)$
5. $id_e \leftarrow GetExitFace(p[0], p[1], p[2], p[3])$
6. $T_{curr} \leftarrow GetNextTet(T_{curr}, id_e, id[id_e])$

OUTPUT The id $T_{curr}$ of the next tetrahedron to visit. The Id $id_e$ of the exit face in the visited tetrahedron. Indices $id[0]$, $id[1]$ and $id[2]$ of the entry face in the current tetrahedron.
```

3. Additional experiments

To evaluate the quality of the rendered images we compute the average PSNR between masked rendered images and masked input images. Note that larger values are then better.

$$PSNR = 20 \log\left(\frac{N_v}{\|I - \tilde{I}\|_2}\right), \tag{1}$$

where N_v is the number of valid pixels as defined by the mask image.

Table 1 shows results of our method, NeuS2 and VOXURF on a subset of the 4D Human dataset [1]. From these results we can see that our proposed method is also extremely efficient at generating high quality appearance models. Our proposed method (VortSDF) obtained a better PSNR than both NeuS2 and VOXURF on all data except f-sho-hx.

Comparison vs. TetraNerF version To further demonstrate the advantage of our proposed densification approach we compare our method with a straightforward extension of TetraNERF [2] with using SDF field. Concretely, we applied our SDF optimization algorithm but with using a fixed

Table 1. Average photometric accuracy $PSNR$ (higher is better) obtained with our method, NeuS 2 and Voxurf, for each of the 8 test scenes.

Method	f-cos-hx	f-jea-hx	f-opt1-hx	f-opt2-hx	f-opt3-hx	f-sho-hx	m-jea-hx	m-opt-hx	avg
NeuS2	27.6	28.4	31.9	33.2	30.3	31.6	29.7	25.8	29.76
VOXURF	28.38	29.32	39.26	37.34	32.82	**36.73**	34.45	37.57	33.23
VortSDF (our)	**28.80**	**29.90**	**39.64**	**37.48**	**32.95**	36.46	**34.71**	**37.86**	**34.72**

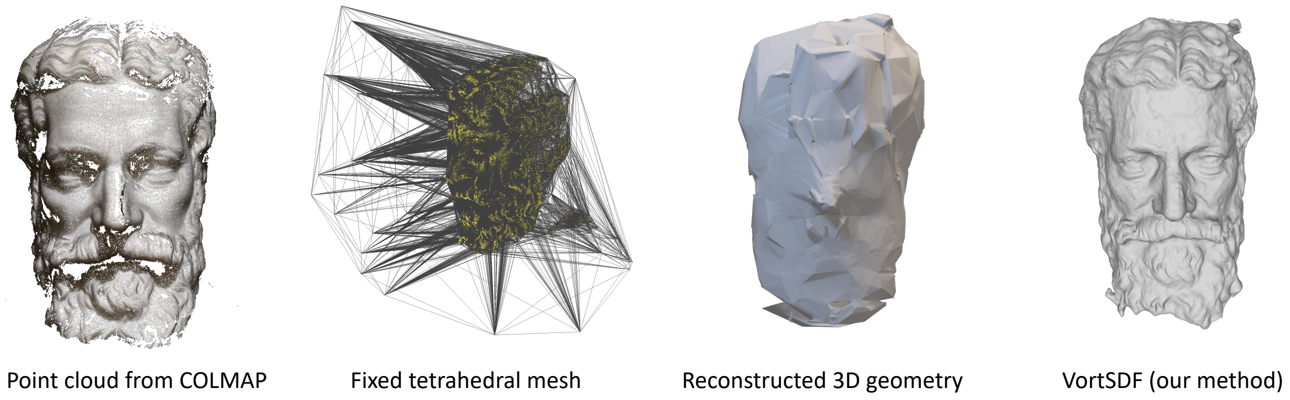

Figure 1. Comparative results we obtained with our method and with using a fixed tetrahedral mesh defined by the dense output of COLMAP.

tetrahedral discretization that is given by COLMAP point cloud. Figure 1 shows that without carefully discretizing the 3D space around the surface a detailed 3D geometry cannot be reconstructed. Only appearance can be modeled.

References

[1] Matthieu Armando, Laurence Boissieux, Edmond Boyer, Jean-Sebastien Franco, Martin Humenberger, Christophe Legras, Vincent Leroy, Mathieu Marsot, Julien Pansiot, Sergi Pujades, Rim Rekik, Gregory Rogez, Anilkumar Swamy, and Stefanie Wuhrer. 4dhumanoutfit: a multi-subject 4d dataset of human motion sequences in varying outfits exhibiting large displacements. *Computer Vision and Image Understanding*. 2

[2] Jonas Kulhanek and Torsten Sattler. Tetra-nerf: Representing neural radiance fields using tetrahedra. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2