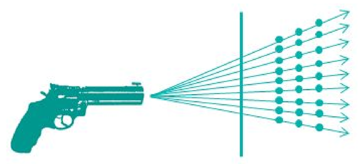
ReVolVR – Rendering Volume Data in VR using HTC Vive



Sebastian Suder
sebastian.suder@alumni.fh-aachen.de

Stefan Schiffer
s.schiffer@fh-aachen.de

Ingrid Scholl
scholl@fh-aachen.de

Mobile Autonomous Systems & Cognitive Robotics Institute (MASCOR)
FH Aachen University of Applied Sciences, Aachen, Germany

MASCOR
Mobile Autonomous Systems and Cognitive Robotics
FH AACHEN UNIVERSITY OF APPLIED SCIENCES

Introduction

Direct Volume Rendering (DVR) techniques are used to visualize surfaces from 3D volume data sets without computing a 3D geometry. Several surfaces can be classified using a transfer function (TF) by mapping data values to color and opacity (RGBα). To find a good transfer function is in general a manual and time-consuming procedure and requires detailed knowledge of the data and the imaging technique. In this poster, a new *Virtual Reality (VR)* application ReVolVR is presented. It is based on the HTC Vive VR technique to render and interact with volume data. ReVolVR loads, modifies and saves the TF in real-time while continuously rendering the stereoscopic 3D volume through GPU-based ray casting shader.

Ray Casting Volume Rendering

Ray Casting:

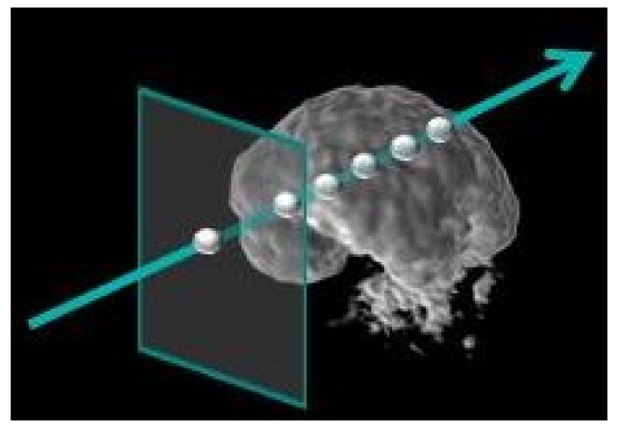
> Ray Casting is a direct volume rendering method.
> Ray Casting traces rays from the camera into the volume and uses sample values along the ray to compute a volume-rendering integral.
> GPU based ray casting parallelize the computations for each ray [3, 1].

Ray Casting Improvements:

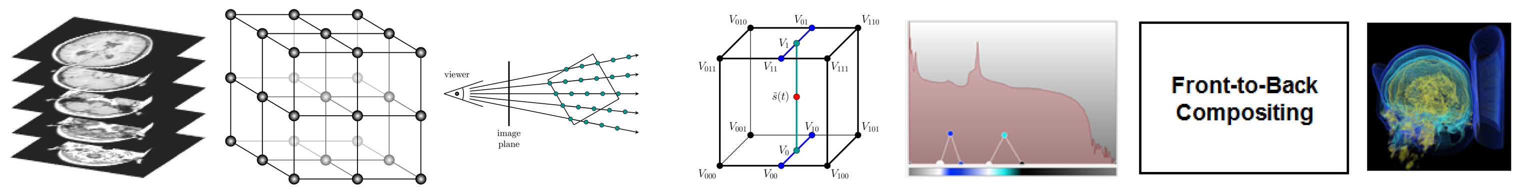
> Empty space skipping.
> Early ray termination: ray traversal can be stopped when the $opacity \geq 1.0$.
> To avoid aliasing jitter the entry point from the ray through the fragment pixel.

Algorithm 1: Pseudocode for single-pass ray casting with pre-integration

```
Determine volume entry position
Compute ray line and direction
Apply stochastic jittering to ray start position along ray direction
while (Current ray position in volume) AND (opacity < 1.0) do
    Access scalar value at current position from volume data
    Classify scalar value with 2D pre-integrated transfer function
    Compositing of color and opacity
    Calculate next sample position along the ray
end
```



Ray Casting Pipeline



DICOM data set | Voxel data set | Sampling $s(t)$ | Trilinear Interpolation $\tilde{s}(t)$ | Classification Transfer Function $\tilde{s}(t) \rightarrow (C_{src}, \alpha_{src})$ | Compositing $(C_{src}, \alpha_{src}) \rightarrow (C_{dst}, \alpha_{dst})$ | Rendering (C_{dst}, α_{dst})

Why we use the GPU?

Ray Casting Complexity Example:

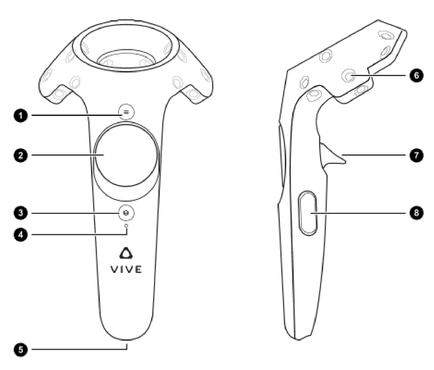
> Volume Data Size 256^3 voxel.
> HTC display resolution 2160×1020 pixel.
> Sample rate per ray $< 256 \cdot \sqrt{3}$.
> Approx. 1.15 billion ops per user pose.
> With 24 fps we need over 27.6 billion ops.

Use GPU architecture:

> Implement all needed algorithms with shaders.
> Using shaders for the parallel raycasting.
> Real-time rendering on every interaction.
> Use high-performance computer with Nvidia GeForce GTX 1080

Volume Interaction

Several user interactions are implemented to pick, translate and scale the volume data in the virtual scene. All interactions are designed to conveniently reflect to real movements of the user as much as possible.



HTC Vive controller [2]

Head and Controller Tracking:

> The Vive base station tracks sensors from the headset and controllers.
> Calculate new transformation matrices with sensor tracked data.
> Apply the transformations and render new stereoscopic image pairs.

Clipping planes:

> Render three different clipping planes (transverse / sagittal / coronal planes) from the original voxel data.
> Toggles on/off the clipping plane rendering.
> Picking only one clipping plane with one controller.
> Sliding the selected clipping plane by moving the picking controller.

Volume Transformations:

> Toggles on/off the volume rendering.
> Pick the volume through intersection the volume with one controller.
> Translate the volume through movements of the picking controller.
> The volume can be scaled by movement of both controller.

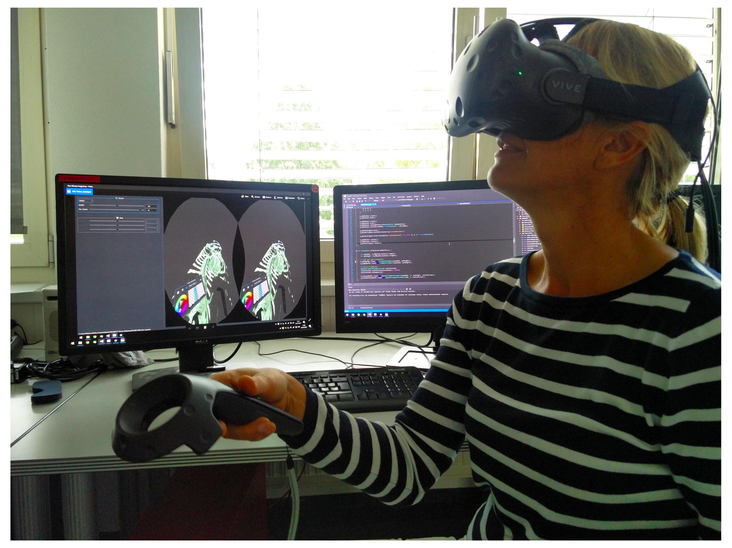
Transfer Function Editor:

We use our GUI design for the 1D transfer function (TF) editor (data range on x-axis, opacity value on y-axis) [4]. Mapping the GUI image as a texture into the virtual scene enables the following TF interactions:

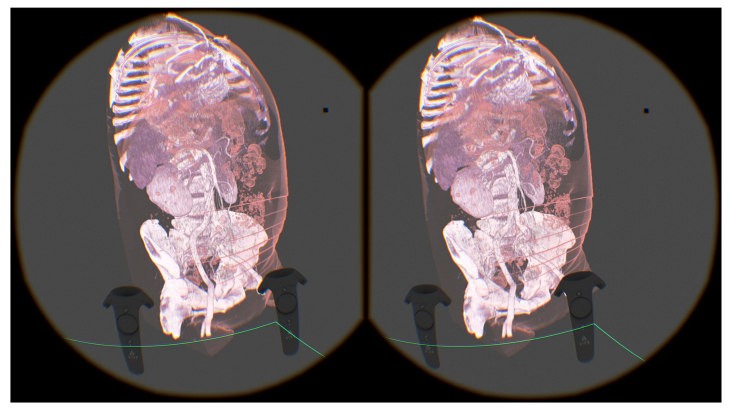
> Toggle on/off the transfer function editor.
> Load predefined transfer functions.
> Design interactively a new transfer function (set color and opacity ranges, define tent functions).
> Render the volume in real-time with new visual properties in the virtual scene.
> Save the current transfer function.

Usability:

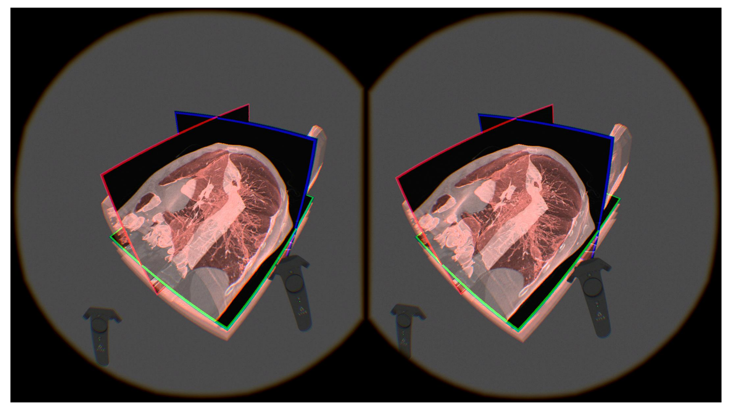
> In a preliminary evaluation run ReVolVR was received very well.
> Users felt it was very intuitive to use, even for technical novices.



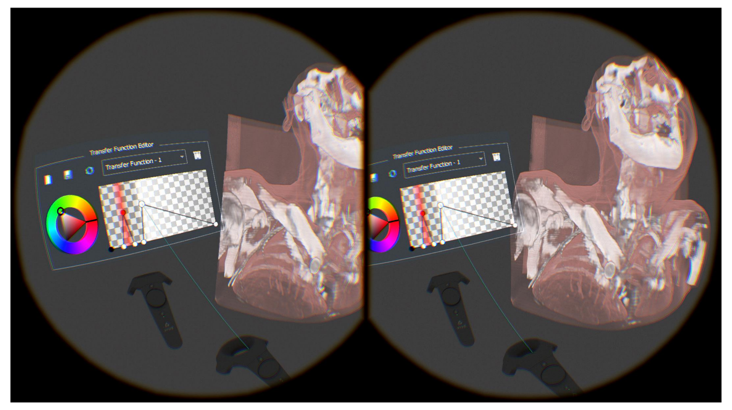
Real scene using the VR application.



Stereoscopic rendering of the ray casting volume.



Blending ray casting volume and clipping planes.



Interactive editing of the transfer function.

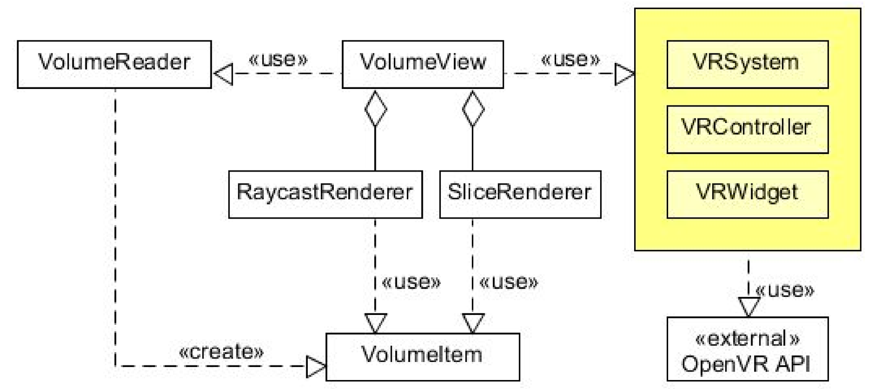
Application Architecture

Implementation Techniques:

> C++, OpenGL 4.5, GLSL 4.5
> APIs: Qt 5.9, OpenVR 1.0.9, DICOM-Toolkit (DCMTK) 3.6.2

VolumeView class:

> Initialize the OpenGL context.
> Render the volume and clipping planes to HTC Vive displays.



Software Architecture from ReVolVR.

VolumeReader class:

> Read DICOM or RAW data files
> Create VolumeItem by adding the correct transformation matrix to the volume data.

VR classes:

> Calculate new transformation matrices from tracked head and controller movements.
> Add the transformations to the VolumeItem and render again.

Summary and Conclusion

Summary:

> Developed VR application for volume rendering with the HTC Vive.
> Intuitive interaction with the volume and clipping planes.
> Interactive editing of the transfer function.
> Real-time rendering using GPU based shader algorithms.

Outlook:

> Integrate shading and optimize interactive transfer functions editing.

Further Information:

> some (stereoscopic) example videos can be found on `http://maskor.fh-aachen.de/projects/ReVolVR/`

Acknowledgement

We would like to acknowledge the Department of Diagnostic and Interventional Radiology, Nils Krämer and Andreas Ritter, and the Department of Oral and Maxillofacial Surgery, Alexander Bartella, Hannes Bothung and Frank Hölzle, from RWTH Aachen University Hospital, Germany, for the feedback and discussions, and supporting us with selected clinical data material.

References

[1] Engel, K., Hadwiger, M., Kniss, J., Rezk-Salama, C., Weiskopf, D.: Real-time volume graphics. CRC Press (2006)

[2] HTC: Vive PRE User Guide. `http://www.htc.com/managed-assets/shared/desktop/vive/Vive_PRE_User_Guide.pdf` (2017), [Online; accessed 27-July-2017]

[3] Krüger, J., Westermann, R.: Acceleration techniques for gpu-based volume rendering. In: Proceedings of the 14th IEEE Visualization 2003 (VIS'03). pp. 38–. VIS '03, IEEE Computer Society, Washington, DC, USA (2003), `http://dx.doi.org/10.1109/VIS.2003.10001`

[4] Schubert, N., Scholl, I.: Comparing gpu-based multi-volume ray casting techniques. Computer Science-Research and Development 26(1), 39–50 (2011)