t-SNE based Transfer Functions for Multi-attribute Volume Rendering

Ravi Snellenberg*
TU Delft

Thomas Höllt†
TU Delft

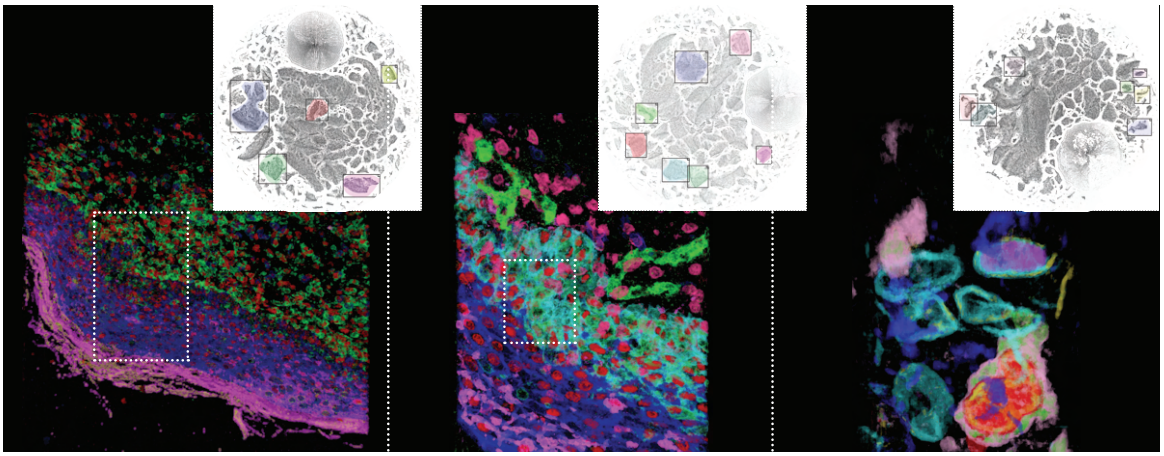


Figure 1: **Volume renderings** of various challenge dataset ROIs at different levels of detail using t-SNE based transfer functions.

1 Introduction

Standard direct volume rendering (DVR) methods are built for scalar data, such as medical scans. While the rendering algorithm extends to multi-dimensional data straightforwardly, the definition of transfer functions (TFs), mapping the input data (domain) to visual properties (co-domain, typically RGBα), is a key challenge.

User interfaces for the design of TFs with a 1D or 2D domain are well established [4], however, designing TFs for higher dimensional domains, such as CycIF, is less explored. A relatively simple approach is to assign color hues to individual dimensions and use the intensity to define opacity. This approach provides only limited freedom and the results of mixed colors are hard to interpret. Explicit multi-dimensional TF design is often based on multi-dimensional data visualizations such as parallel coordinate plots [8] or star coordinate plots [3]. These methods, however, work best with no more than 10 to 20 dimensions.

Another approach for TF design is using dimensionality reduction (DR) to reduce the data to 2D and use common 2D TF interfaces. This works well, and has been applied [2] with linear DR, such as principal component analysis (PCA). In the (spatial) single-cell field, however, PCA has been replaced by neighborhood-preserving DR techniques, such as t-SNE or UMAP, as they allow easy identification of different cell types in the low-dimensional embedding. However, due to their non-linear nature, these methods do not allow for easy lookup of new values, such as those created by interpolated sampling during DVR, in the existing embedding.

We designed a t-SNE-based transfer function (TF), along with several volume rendering methods and apply it to the Bio+MedVis 2025 challenge CycIF data. In this work we focus on one method, providing very high visual fidelity. While this method comes with rather slow rendering times, combining it with another high-performance method during interaction provides the possibility for deployment in interactive visualization systems.

*e-mail: r.snellenberg@student.tudelft.nl
†e-mail: t.hollt-1@tudelft.nl

2 Volume Rendering Multi-attribute Data

Direct volume rendering (DVR), in brief, works in the following way; **1.** For every pixel on screen, a ray is cast into the volume. **2.** Along the ray, samples of the volume are taken and mapped to color and opacity according to a user-defined TF. **3.** The color of all samples is then aggregated to define the color of the pixel.

2.1 Transfer Function

In this work, we aim to use t-SNE embeddings to guide the creation of TFs used in **2**. We use a common 2D TF view (Figure 2) with a t-SNE embedding as background to guide the user (Figure 2a). The user can place widgets (Figure 2b) on the 2D domain and assign color and opacity using a separate panel (Figure 2c). In addition to the standard TF, we provide a material-based TF to highlight material boundaries [1]. The user can define visual properties individually, depending on whether the ray is entering and leaving a material (Figure 2d).

2.2 Linear Interpolation

As samples are typically not taken on voxel centers, step **2.** requires some form of interpolation. Linear interpolation is commonly used, providing a good tradeoff between visual quality and speed, and common low-dimensional TFs use a linear domain. However, any form of interpolation, beyond nearest neighbor interpolation, creates values that are potentially not in the input set. However, t-SNE (like other non-linear DR methods) does not directly support the projection of new points, meaning that looking up an interpolated value requires some form of approximation.

Here, we approximate the projection of an interpolated sample by identifying a number of nearest neighbors (NNs) in the high-dimensional space and look up their position in the two-dimensional embedding, similar to how e.g., openTSNE initializes new datapoints in an existing embedding before further optimizing [6]. As t-SNE and similar methods optimize for similarity, we can expect that the identified points would also be close in the 2D space. A single NN often produces good results, but when using more than one NN, we filter out the outliers before we apply a weighted average operation, as the DR is not always able to put all points that are supposed to be in one cluster together in the lower-

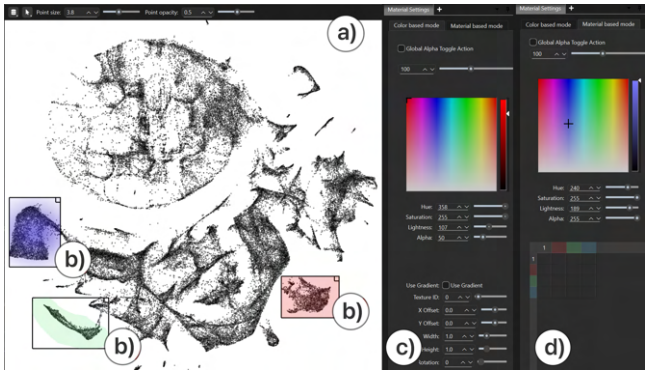


Figure 2: **The transfer function widget interface.** a) t-SNE embedding, b) TF widgets, c) and d) various widget settings.

dimensional space. The resulting position for the interpolated sample can then be used to look up the color and opacity in the TF.

2.3 Approximate Nearest Neighbors

Calculating the exact NNs for every sample is costly. Therefore, we propose to use approximated nearest neighbors (ANNs), as has also become the de facto standard for calculating similarities in neighborhood-based DR, such as t-SNE [5]. The choice of the ANN algorithm does have a noticeable effect on the correctness of the returned positions. Unfortunately, the most accurate ANN algorithms that would allow their use directly in a custom shader on the GPU require restrictive hardware and driver support. Furthermore, GPUs typically have limited VRAM, which would be further constrained by the need to store auxiliary graph structures alongside large multivariate datasets. Therefore, we implement a mixed CPU/GPU pipeline, described in the following section.

2.4 ANN Rendering Pipeline

Figure 3 shows the ANN rendering pipeline. The volume rendering loop, including sampling and compositing, runs on the GPU, and the nearest neighbor querying is performed on the CPU. To support this mixed architecture sampling, ANN querying, and final compositing are executed in separate passes. First, all samples along rays are collected, then ANNs are queried, returning a position in TF space per sample, and finally, samples are classified based on the gathered position and composited.

Due to memory limits on typical GPUs, storing all intermediate results in a single pass is often not feasible. For instance, assuming the volume occupies the entire screen, rendering a 70-dimensional dataset (32-bit float) to a full HD screen at an average of 50 samples per ray would require roughly 29GB of memory. Therefore, we batch the rays across the used screen space, allowing partial results to be displayed progressively while keeping memory usage bounded. One of the downsides of the staged approach is that features depending on sample output, like early ray termination, are not possible. However, collecting all samples first allows us to calculate the required memory beforehand to optimize batch sizes.

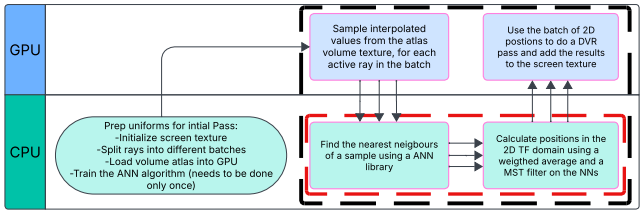


Figure 3: **The ANN Data Render Pipeline.** Everything within the black dotted line is done once per batch. Everything within the red dotted line is done in parallel.

Once retrieved, sampled points are processed on the CPU, where the corresponding 2D positions are retrieved and the weighted average is calculated. Finally, we upload the resulting 2D positions back to the GPU, where they can be classified using the TF and composited using a compositing scheme. The only modification that is needed compared to the standard scheme is the sampling step. As we need to sample from the passed buffer containing the 2D positions of the processed batch of rays, instead of the volume.

We have implemented the described pipeline in ManiVault Studio [7], where it can be combined with other visualizations, e.g., to show the feature space of clusters in a heatmap to aid interpretation of biological meaning of selected structures.

3 Discussion and Conclusion

We have presented a volume rendering pipeline for multi-attribute volume data, using 2D TFs based on the widely used t-SNE DR. The presented volume rendering pipeline produces high-quality output. However, even with approximation, the NN lookup is prohibitively expensive for real-time applications. A single frame can take from seconds to minutes. We have investigated a number of other approaches (see Supplemental Figure 3), including simple pre-classification (assigning colors to voxels directly and interpolating those) and directly interpolating the points in the 2D domain. Generally, these methods can be used for real-time visualization, but provide inferior image quality. We propose a combination, where one of the faster render modes is used during interaction and the result is replaced by the ANN pipeline once ready. Possible future work could be adding an automatic TF, for example, by clustering the resulting t-SNE DR and converting clusters to TF-widgets. Another open issue is addressing the extreme scale of some of this data. We are considering hierarchical DR to improve scalability, both in the computation and visual space of the DR.

References

[1] O. Igouchkine, Y. Zhang, and K.-L. Ma. Multi-material volume rendering with a physically-based surface reflection model. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3147–3159, 2018. doi: 10.1109/TVCG.2017.2784830 1

[2] H. S. Kim, J. P. Schulze, A. C. Cone, G. E. Sosinsky, and M. E. Martone. Dimensionality reduction on multi-dimensional transfer functions for multi-channel volume data sets. *Information Visualization*, 9(3):167–180, 2010. doi: 10.1057/ivs.2010.6 1

[3] A. Kumar, X. Zhang, H. L. Xin, H. Yan, X. Huang, W. Xu, and K. Mueller. Radvolviz: An information display-inspired transfer function editor for multivariate volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 30(8):4464–4479, 2024. doi: 10.1109/TVCG.2023.3263856 1

[4] P. Ljung, J. Krüger, E. Groller, M. Hadwiger, C. D. Hansen, and A. Ynnerman. State of the art in transfer functions for direct volume rendering. *Computer Graphics Forum*, 35(3):669–691, 2016. doi: 10.1111/cgf.12934 1

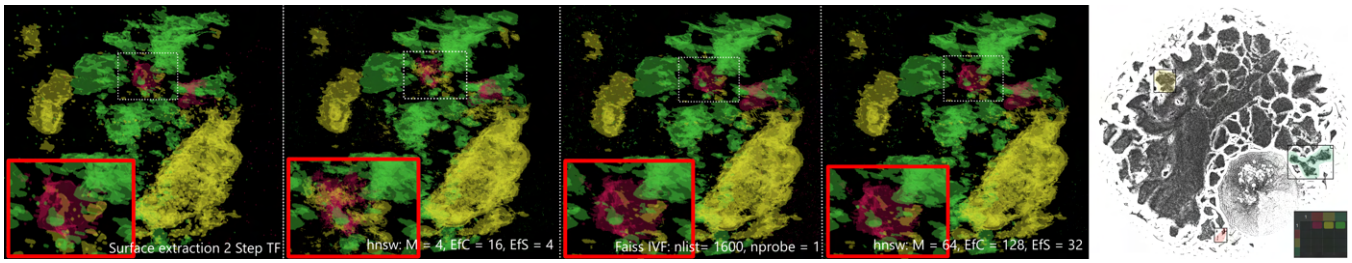
[5] N. Pezzotti, B. P. F. Lelieveldt, L. v. d. Maaten, T. Hollt, E. Eisemann, and A. Vilanova. Approximated and user steerable tSNE for progressive visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1739–1752, 2017. doi: 10.1109/tvcg.2016.2570755 2

[6] P. G. Policar, M. Strazar, and B. Zupan. openTSNE: A modular python library for t-SNE dimensionality reduction and embedding. *Journal of Statistical Software*, 109(3), 2024. doi: 10.18637/jss.v109.i03 1

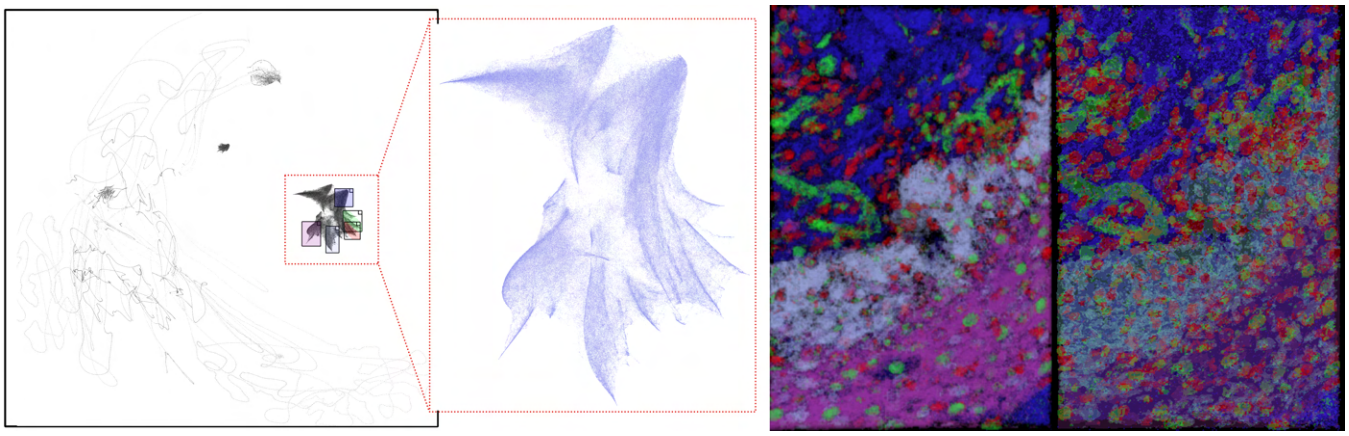
[7] A. Vieth, T. Kroes, J. Thijssen, B. van Lew, J. Eggermont, S. Basu, E. Eisemann, A. Vilanova, T. Höllt, and B. Lelieveldt. Manivault: A flexible and extensible visual analytics framework for high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):175–185, 2024. doi: 10.1109/TVCG.2023.3326582 2

[8] L. Zhou and C. Hansen. Transfer function design based on user selected samples for intuitive multivariate volume exploration. In *2013 IEEE Pacific Visualization Symposium (PacificVis)*, p. 73–80. IEEE, 2013. doi: 10.1109/pacificvis.2013.6596130 1

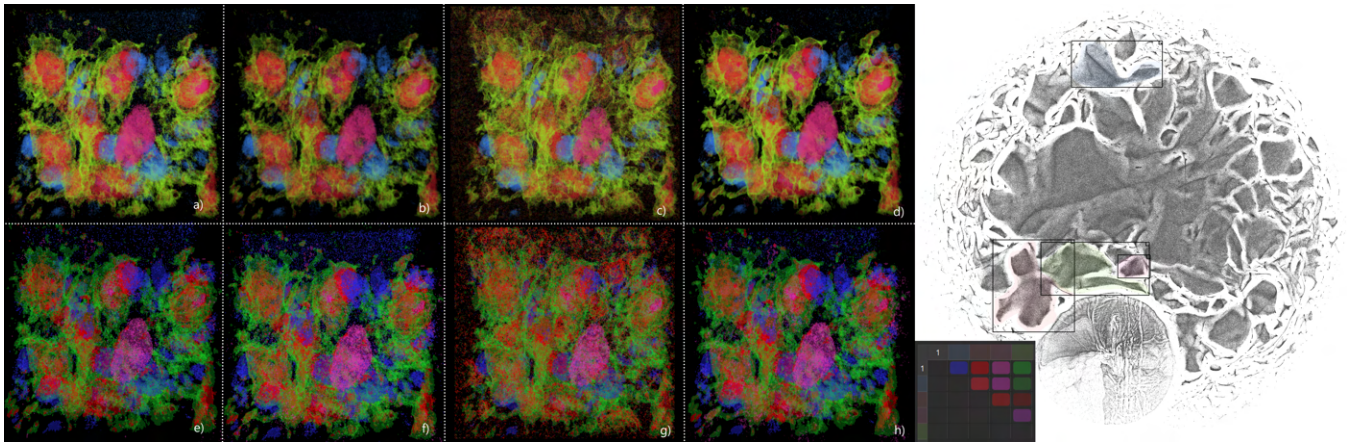
4 Supplemental Figures



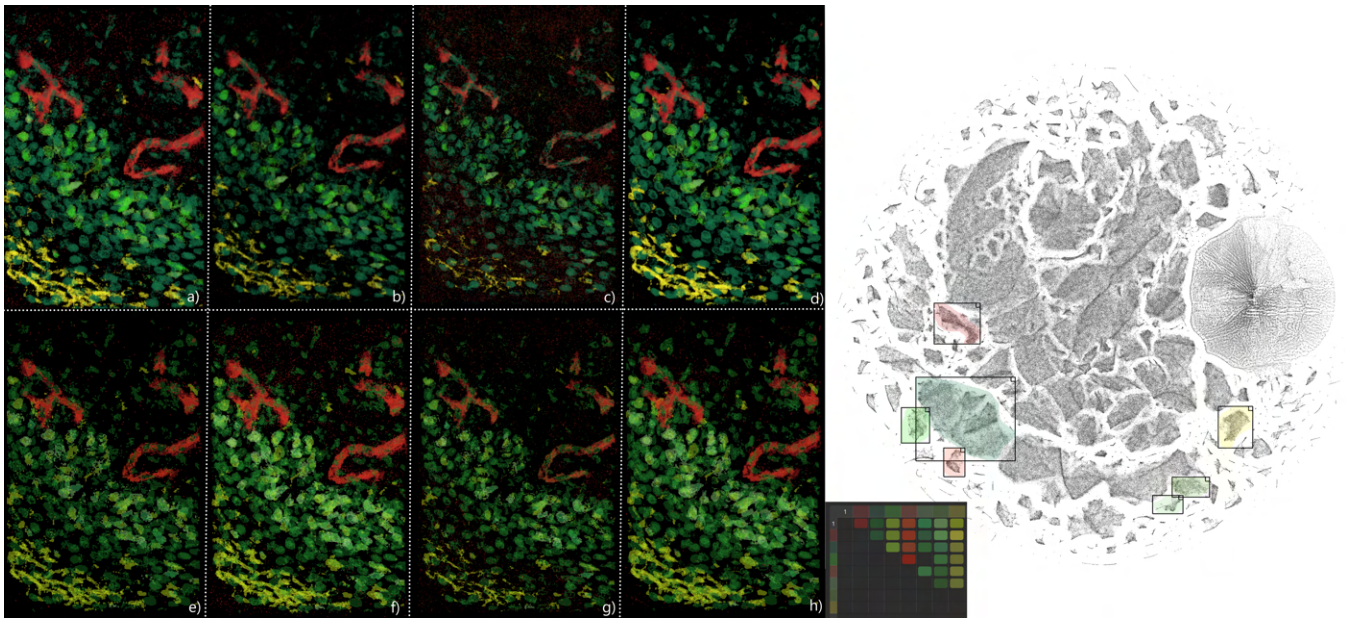
Supplemental Figure 1: Comparison of how ANN algorithms with different accuracies affect the visualization



Supplemental Figure 2: Visualization using a UMAP embedding in the TF, showing other neighbourhood-preserving DR algorithms also work



(a) High-resolution slice (compression 1) of the tissue dataset



(b) Medium-resolution (compression 3) slice of the tissue dataset

Supplemental Figure 3: Comparison of all render modes: a) NN sampling with classic TF, b) pre-classification sampling, c) linear interpolation in the embedded space with classic TF, d) full data interpolation pipeline with classic TF, e) NN sampling with material TF, f) NN sampling with integrated surface extraction using material TF, g) linear interpolation in the embedded space with material TF, h) full data interpolation pipeline with material TF. And on the right side the TF used for rendering the examples. More details on the implementation of other modes can be found at https://repository.tudelft.nl/record/uuid:91d45452-416f-4fda-bfb5-261be169f958