Volume Visualization and Rendering

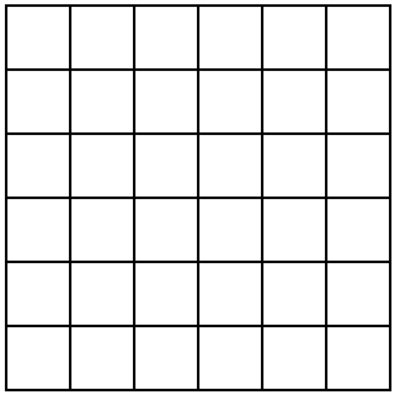
Content

Computer Graphics and Visualization

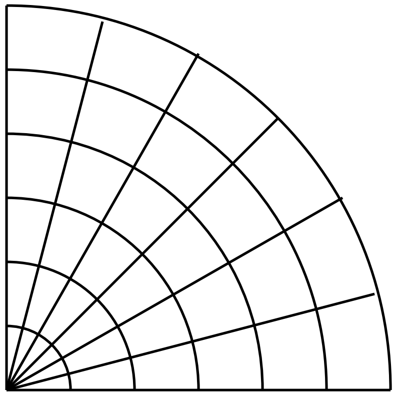
Direct Volume Rendering

RENDERING ALGORITHMS

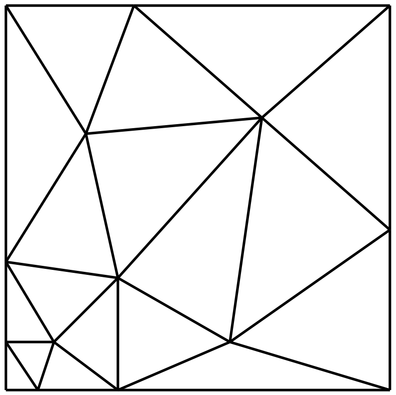
Rendering Algorithms – Overview



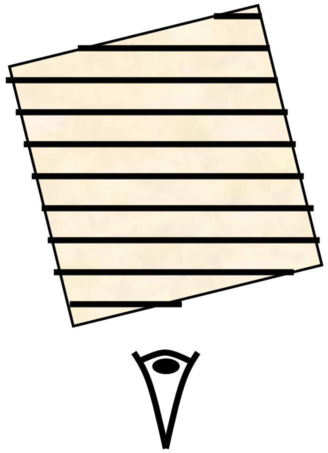
regular grid



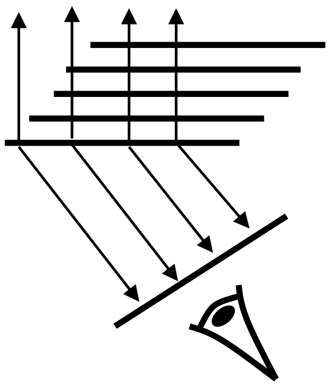
structured grid



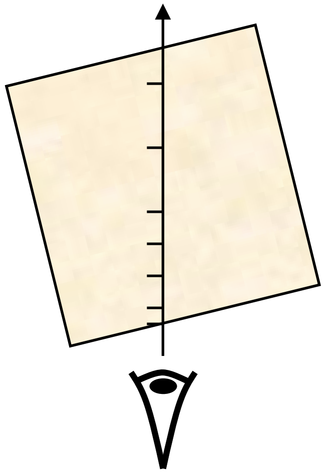
tetrahedral mesh



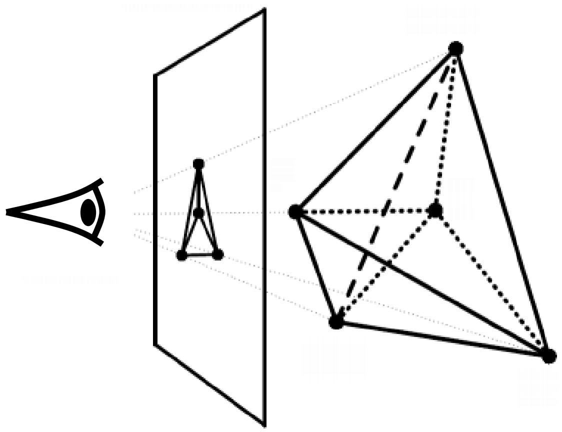
Textur-Slicing



Shear Warp



Raycasting



Projection

Rendering Algorithms – Classification

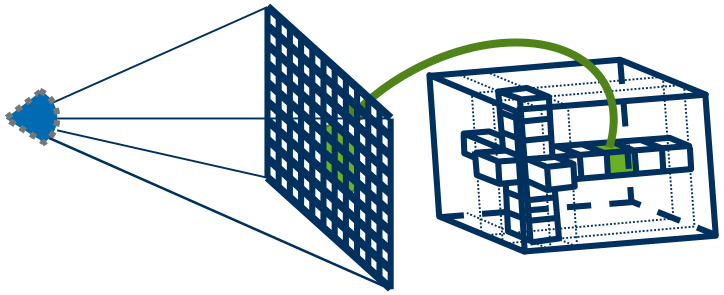
Computer Graphics and Visualization

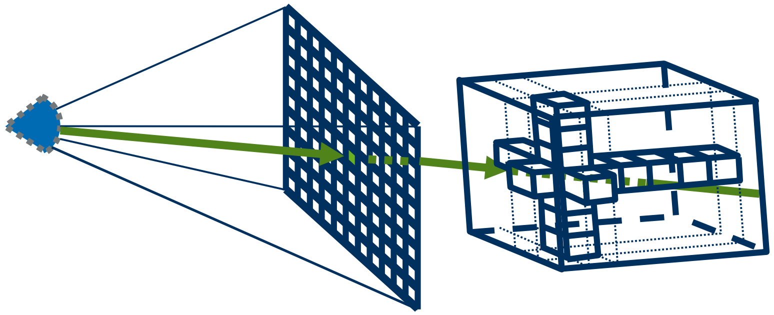
Forward methods

- Object space algorithms
- Cell projection
- Execution: voxel by voxel
- Visibility sorting needed
- Example: slicing, shear-warp, splatting, …

Backward methods

- Image space algorithms
- Execution: pixel by pixel
- Example: Ray casting

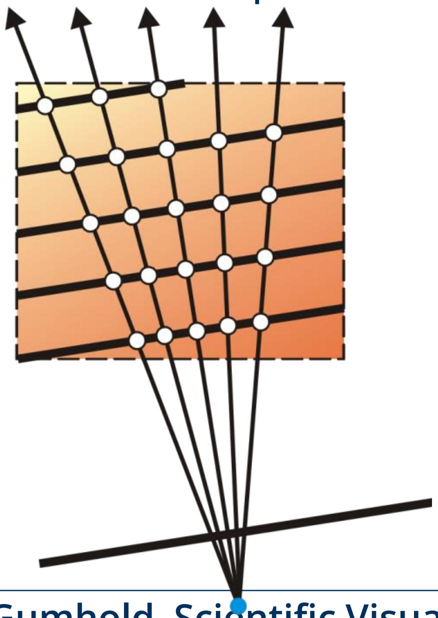


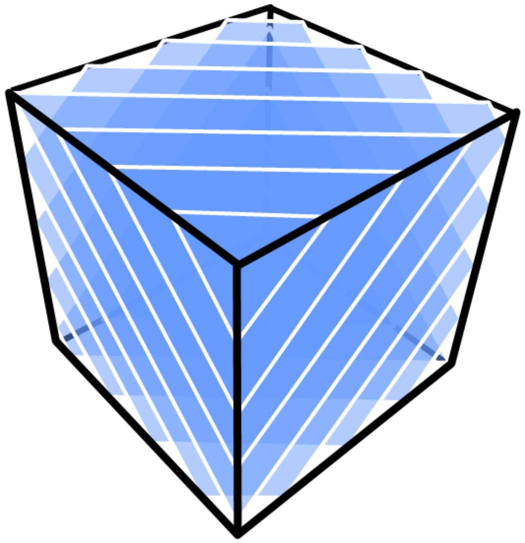


Texture Slicing

- Load voxel grid into 3D texture
- slice texture with planes perpendicular to view direction (back to front or front to back)
- Clipping at volume box
  - enable 6 clipping planes in OpenGL
  - draw quads that cover volume box in screen space

```
// in world space:
wbox = BoundingBox of Volume
for each plane pln of wbox
    enableClipPlane(pln)
// in camera coordinate system:
cbox = BoundingBox of Volume
$z_{min}$ = max($z_{near}$, $min_z$(cbox))
$z_{max}$ = min($z_{far}$, $max_z$(cbox))
for z=$z_{max}$ to $z_{min}$ step -dz do
    render quad at distance z
    that covers cbox
disableClipPlanes()
```





Raycasting

Implementation

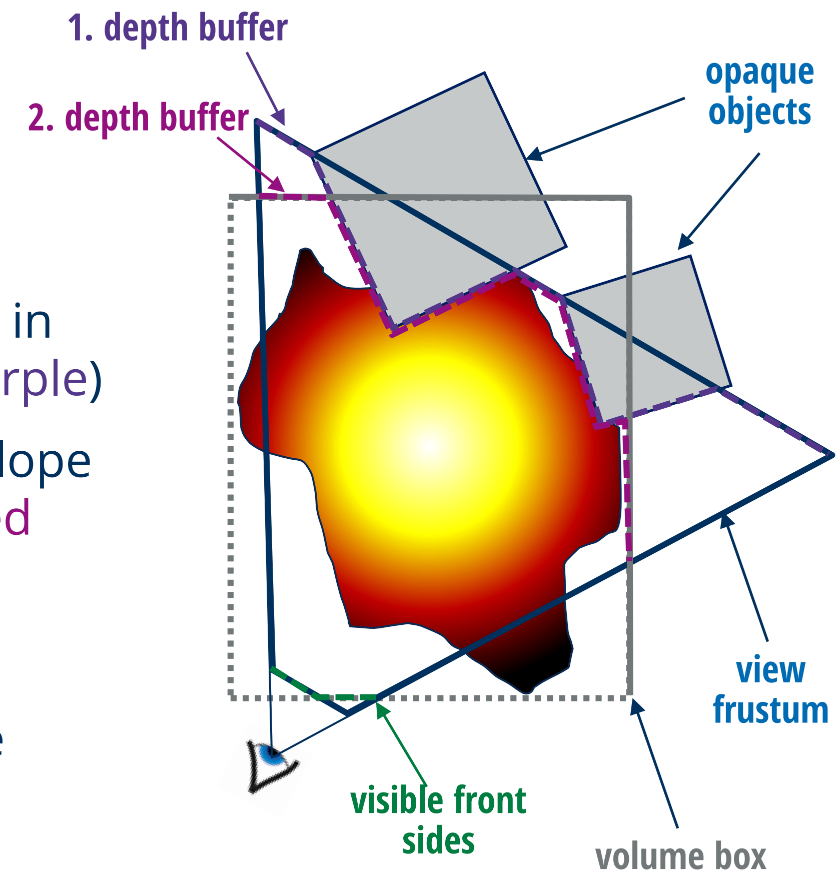
- Raycasting can be implemented in the shader pipeline on the GPU by sampling view ray for each pixel covered by volume box

Preparation:

- Display of all opaque areas with writing in the color and depth buffers (dashed purple)
- Rasterize the backs of the volume envelope with only writing in depth buffer (dashed pink)

raycasting:

- Depth buffer (copy and) bind as texture (dashed pink)
- Rasterize the front sides of the volume envelope with Raycasting shader program and the near-clipping plane (dashed green)



Raycasting – Shader Program

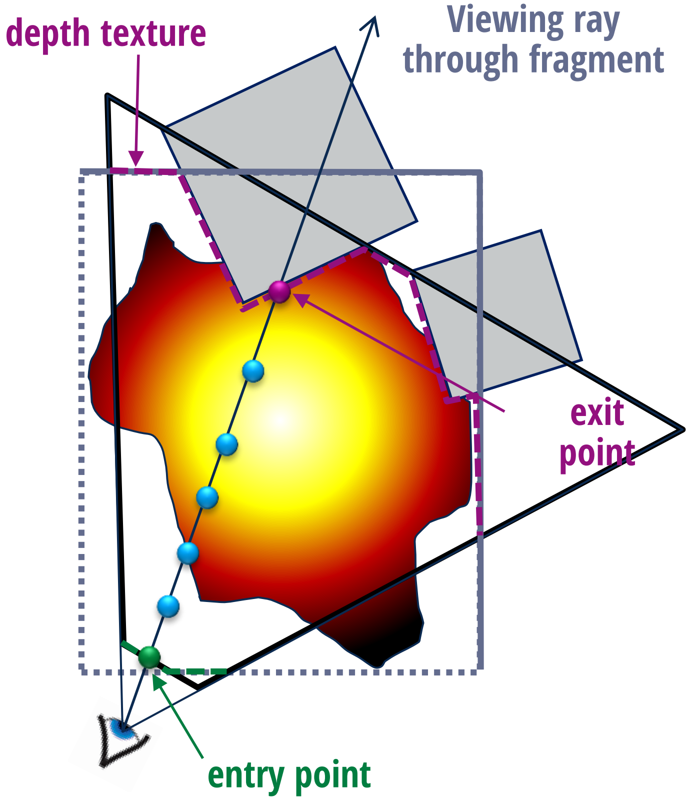
Computer Graphics and Visualization

Vertex shader

- transform position to eye and clip coordinates
- calculate 3D texture coordinates if not part of the geometry data

Fragment shader

- Lookup depth value in depth texture to compute exit point with texture coordinate
- sample ray front to back in while loop
  - get interpolated scalar value from 3D texture and evaluate transfer function
  - blend color and opacity of samples with over operator
  - adapt step width to accumulated opacity
  - remember depth value when opacity threshold was reached first
- Blend accumulated color and opacity over color buffer
- write stored depth value to depth buffer



Raycasting – Acceleration Techniques

Computer Graphics and Visualization

- empty space skipping
- early ray termination
  - Check accumulated opacity for threshold, e.g. 95%
- sampling rate adjustment
  - homogeneity acceleration: the more homogeneous the volume region, the smaller the sampling rate
  - β-acceleration: the higher the accumulated opacity, the smaller the sampling rate

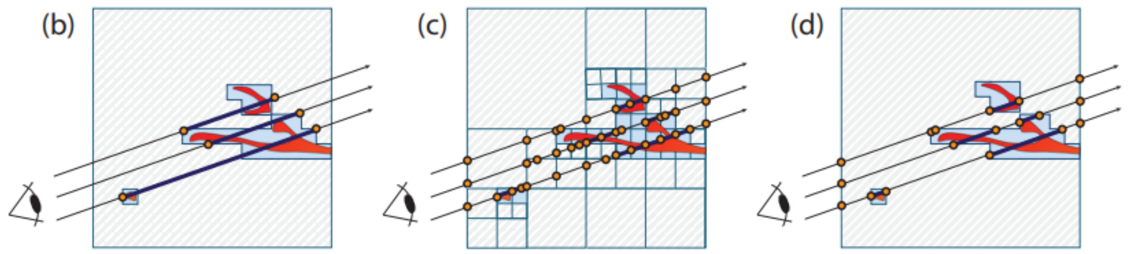
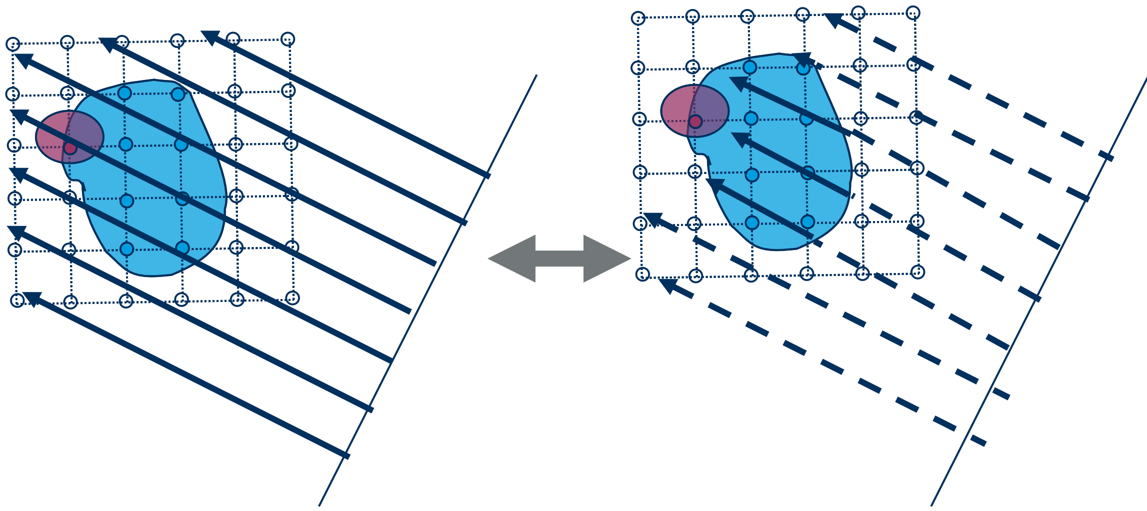


Image from SparseLeap paper from:
b) non-empty bounding geometry rendering, c) octree, d) SparseLeap



left: without, right: with early ray termination

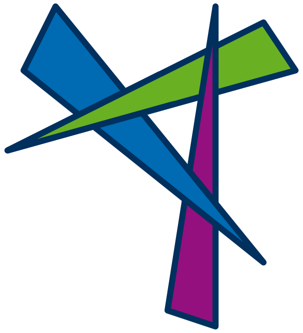
Projected Tetrahedra

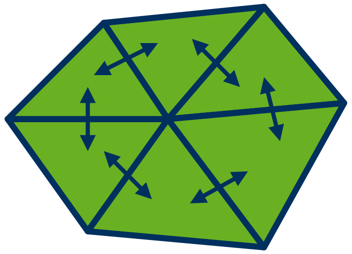
- If data is given on tetrahedral mesh, a cell projection approach called projected tetrahedra is used
- The tetrahedra need to be visibility sorted

Projected Tetrahedra – Sorting

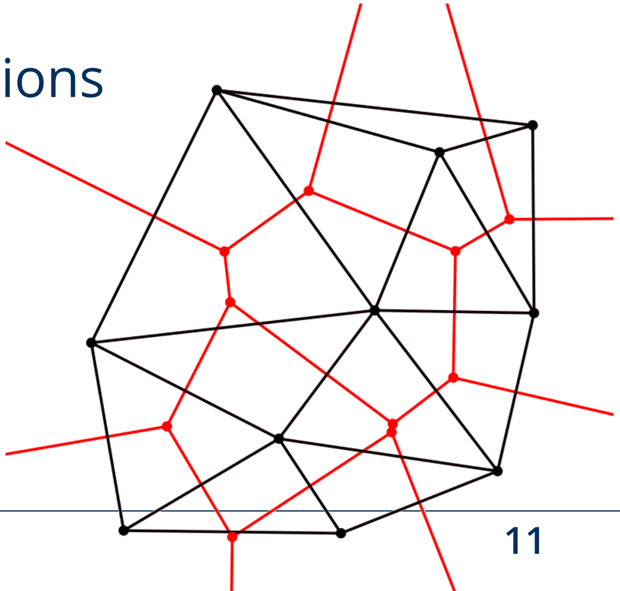
Visibility Sorting Approaches for Tet Meshes

1. Topological Sorting
   - Meshed Polyhedra Visibility Sorting (MPVO)
   - Assumes convex boundary surface
   - Assumes that no visibility cycles can arise
   - builds on tet adjacency data structure





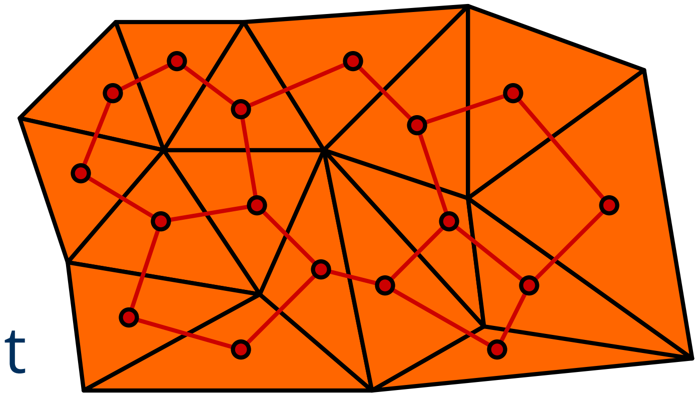
2. Distance Based Sorting
   - Specifically for Delaunay Tetrahedralizations
   - Proveably no visibility cycles exist

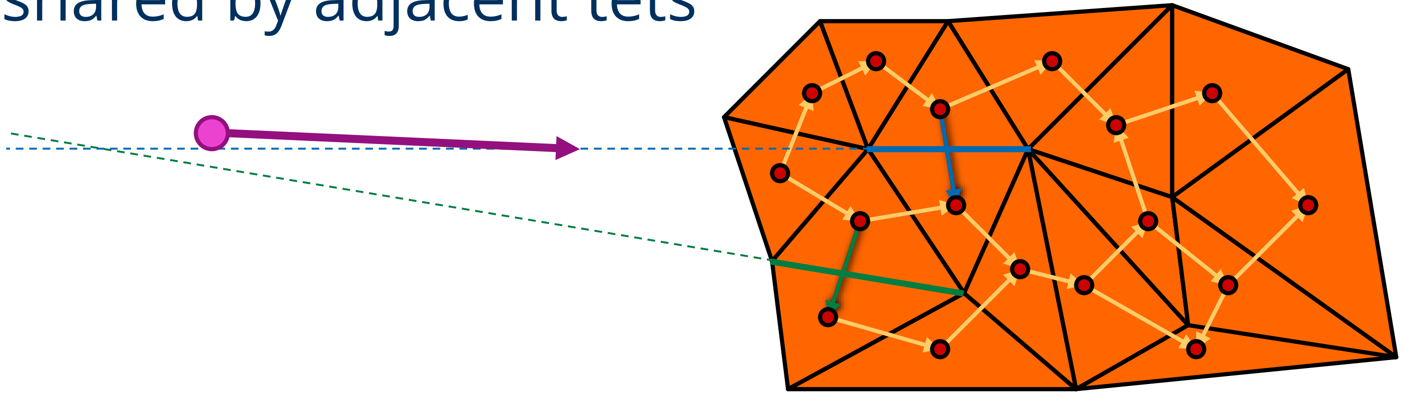


Projected Tetrahedra – Sorting

Meshed Polyhedra Visibility Sorting (MPVO):

- In preprocessing step construct dual graph with one edge per tet-tet-adjacency
- For a given view-point orient each edge based on the side that the view-point lies with respect to the face shared by adjacent tets





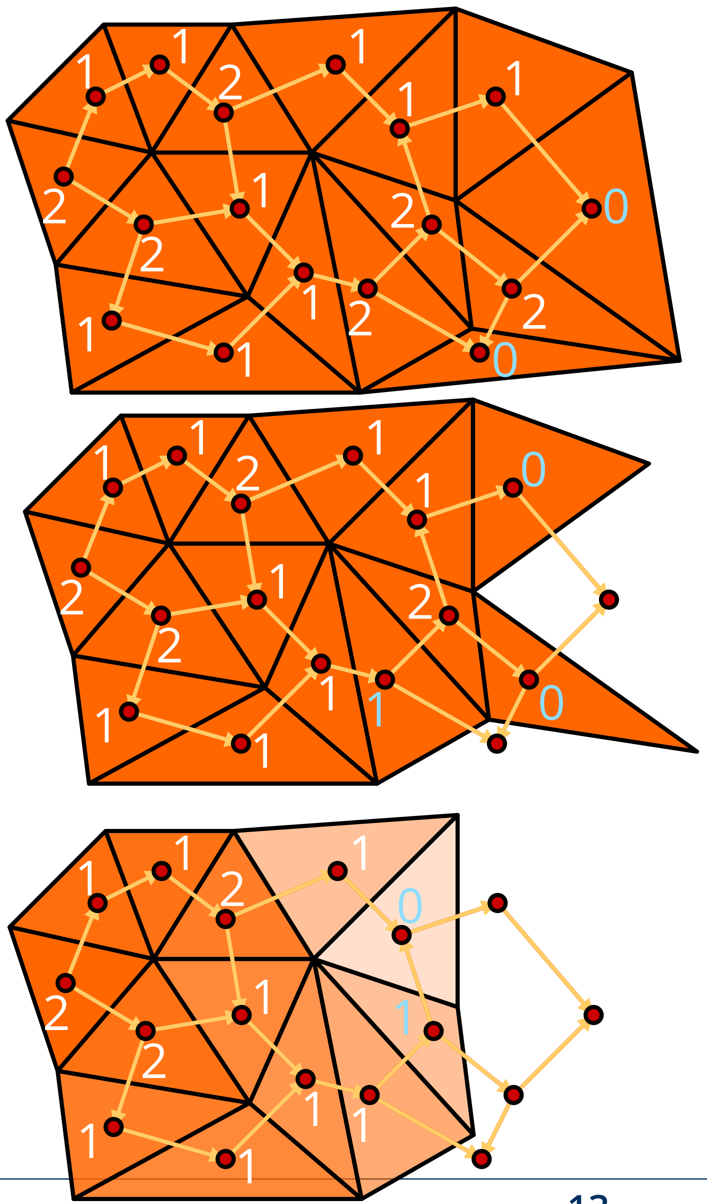
Williams, P. L. (1992). Visibility-ordering meshed polyhedra. *ACM Transactions on Graphics (TOG)*, 11(2), 103-126. (link)

Projected Tetrahedra – Sorting

MPVO continued:

- topologically sort back to front
  - For each tet count number of outgoing edges
  - All tet with zero count do not occlude any other tet and can be removed
  - decrement counts of tets with edge pointing to removed tet
  - Repeat until all tets are removed
- Draw tets in order of removal
- Extension to non convex boundary surfaces:

Callahan, S. P., Ikits, M., Comba, J. L. D., & Silva, C. T. (2005). Hardware-assisted visibility sorting for unstructured volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 11(3), 285-295. (link)



Projected Tetrahedra – Sorting

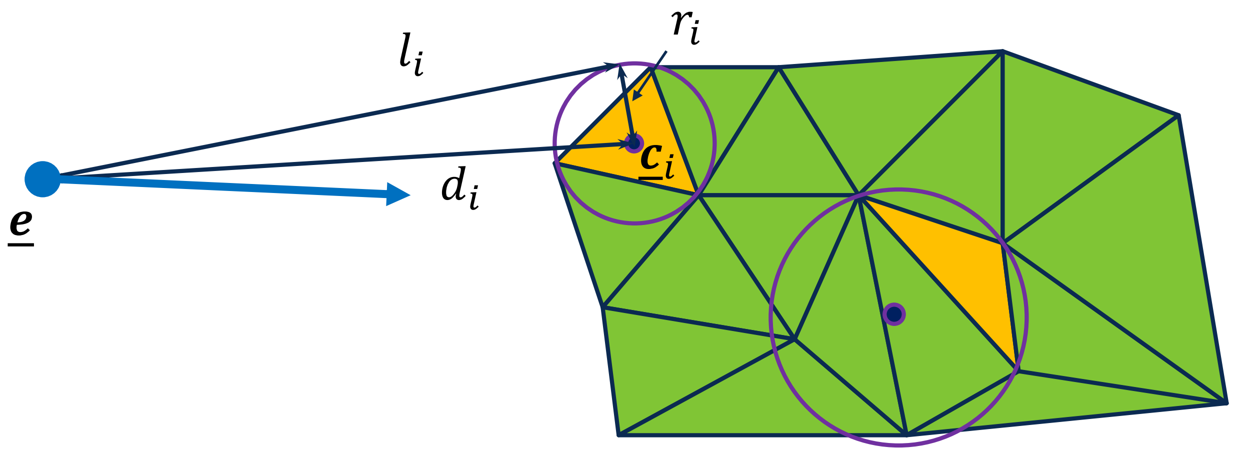
Distance based:

Max, Hanrahan, Crawfis, Area and volume coherence for efficient visualization of 3D scalar fields, ACM Computer Graphics 24(5), 27-33, 1990

- *Assumption:* Delaunay Tetrahedralization
- For each tet compute circum-sphere $(\underline{\boldsymbol{c}}_i, r_i)$
- For view-point $\underline{\boldsymbol{e}}$ compute per sphere distance to silhouette

$$l_i^2 = d_i^2 - r_i^2, \qquad d_i^2 = \left\|\underline{\boldsymbol{e}} - \underline{\boldsymbol{c}}_i\right\|^2$$

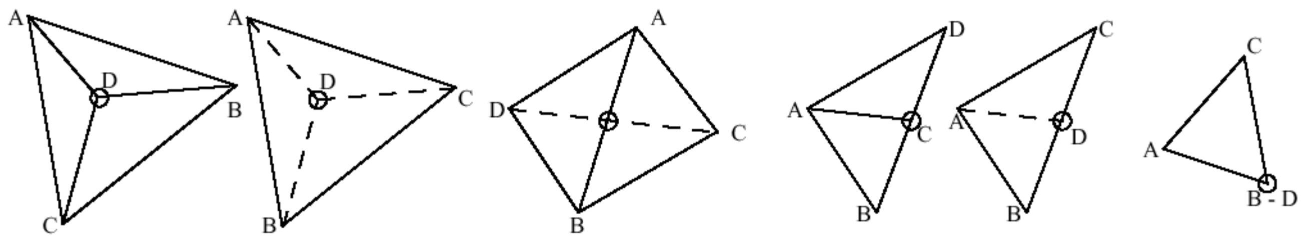
- sort tets by l_i^2



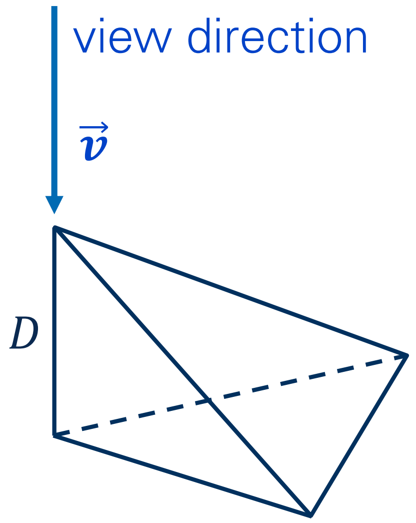
- standard sort like Quick-sort is $O(n \cdot \log n)$, whereas topological sort is $O(n)$

Projected Tetrahedra – Rendering

- For rendering each tet is split view-dependently into up to 4 tets such that each tet protects to a triangle with two corners of zero depth and one corner of depth D.
- Classification into four cases:

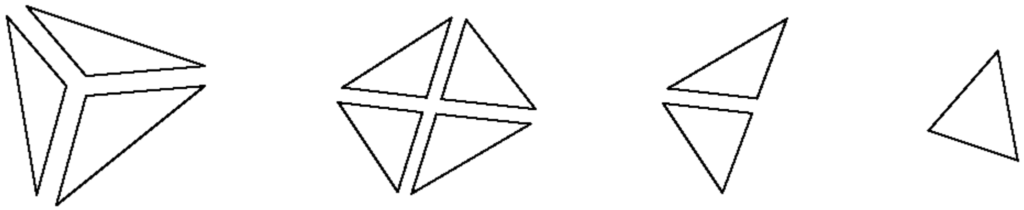


classes 1a & 1b class 2 classes 3a & 3b class 4



view-aligned tetrahedron

- Each resulting tet is rendered as triangle with the depth as attribute:



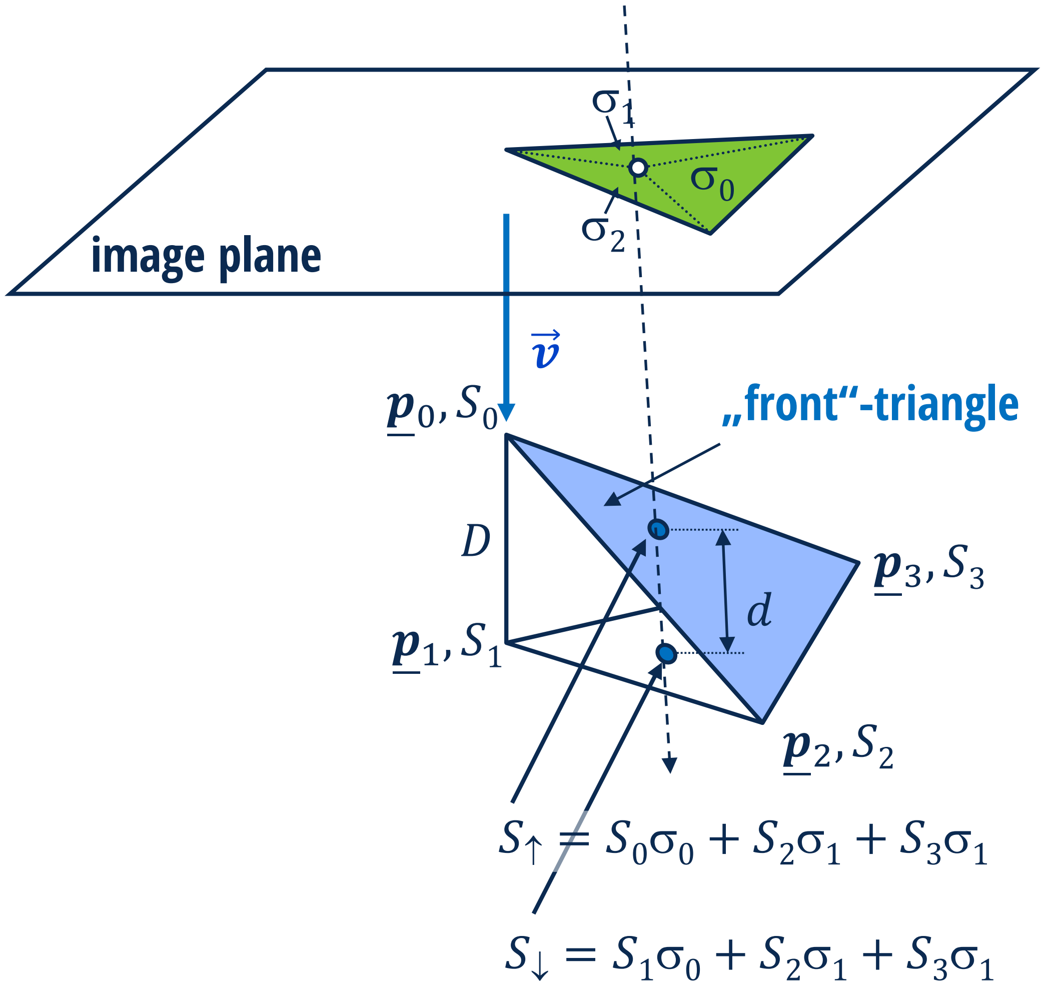
3 triangles 4 triangles 2 triangles 1 triangle

Projected Tetrahedra – Rendering

Computer Graphics and Visualization

Rasterization of view-aligned tets

- rasterize the triangle facing the observer („front"-triangle)
- Interpolate scalar attribute S and eye position $\underline{\boldsymbol{p}}$ linearly on front / back triangle $S_\uparrow$, $\underline{\boldsymbol{p}}_\uparrow$ / $S_\downarrow$, $\underline{\boldsymbol{p}}_\downarrow$
- Lookup volume integral in pre-integration table: $\dddot{\tilde{\boldsymbol{E}}}(S_\uparrow, S_\downarrow, d), T(S_\uparrow, S_\downarrow, d)$ with $d = \left\| \underline{\boldsymbol{p}}_\uparrow - \underline{\boldsymbol{p}}_\downarrow \right\|$

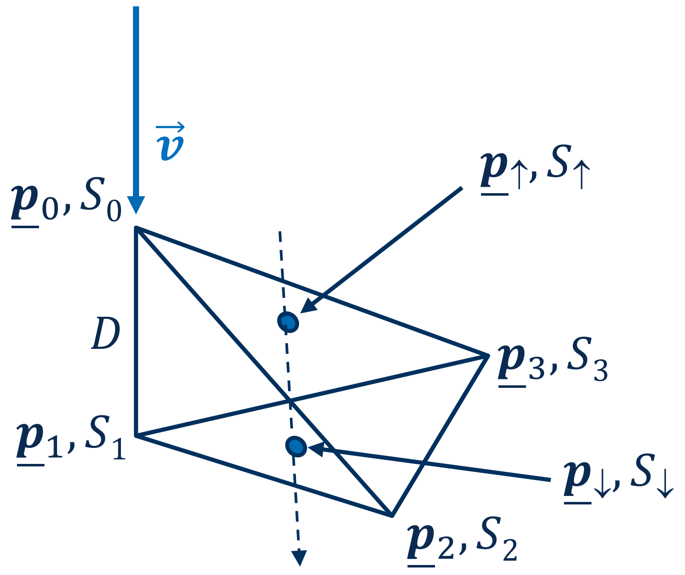


σ_1
σ_0
σ_2
image plane
$\vec{v}$
$\underline{\boldsymbol{p}}_0, S_0$
„front"-triangle
D
$\underline{\boldsymbol{p}}_3, S_3$
d
$\underline{\boldsymbol{p}}_1, S_1$
$\underline{\boldsymbol{p}}_2, S_2$
$S_\uparrow = S_0\sigma_0 + S_2\sigma_1 + S_3\sigma_1$
$S_\downarrow = S_1\sigma_0 + S_2\sigma_1 + S_3\sigma_1$

Projected Tetrahedra – Rendering

Computer Graphics and Visualization

- **Rasterization of view-aligned tets**
- perspectively correct interpolation only works automatically for $\underline{\boldsymbol{p}}_\uparrow, S_\uparrow$ on GPU
- $\underline{\boldsymbol{p}}_\downarrow$ and $S_\downarrow$ need to be multiplied with fraction $w_\uparrow / w_\downarrow$ of $\underline{\boldsymbol{p}}$'s clip space w-components before interpolation (vertex/geometry shader) and corrected in fragment shader by multiplication with $w_\downarrow / w_\uparrow$ of interpolated positions.

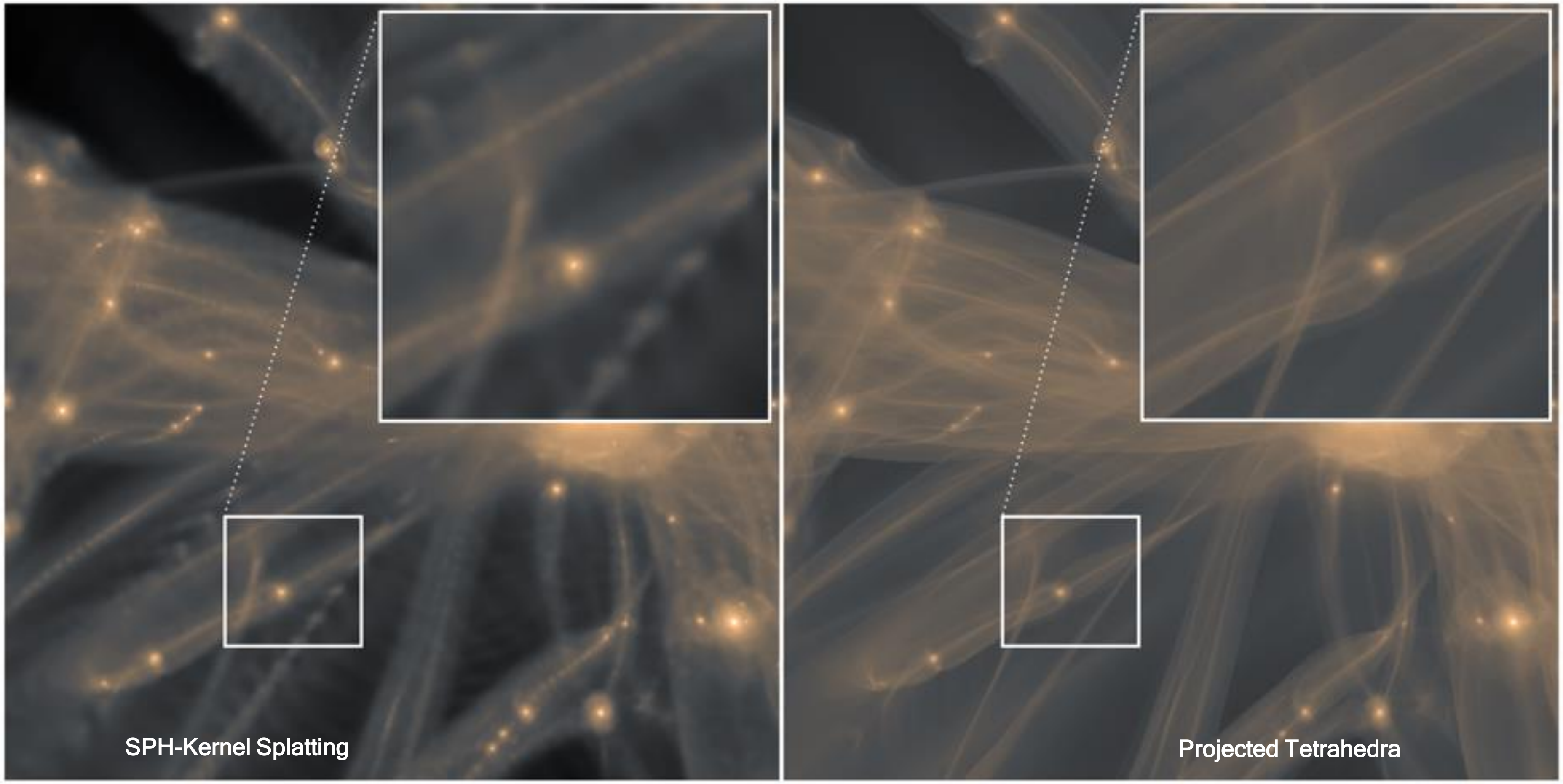


$\vec{v}$
$\underline{\boldsymbol{p}}_\uparrow, S_\uparrow$
$\underline{\boldsymbol{p}}_0, S_0$
D
$\underline{\boldsymbol{p}}_3, S_3$
$\underline{\boldsymbol{p}}_1, S_1$
$\underline{\boldsymbol{p}}_\downarrow, S_\downarrow$
$\underline{\boldsymbol{p}}_2, S_2$

Projected Tetrahedra – Rendering

- In a shader implementation, tetrahedra are drawn with point primitives that are index quadrupels
- The vertex shader looks up the vertex positions and attributes in a texture or shader storage buffer objects
- The geometry shader classifies into the four cases and emits one up to four triangles for the view-aligned tets
- The fragment shader looks up the volume rendering integral in a texture
- The resulting fragment colors are blended in the frame buffer with over operator front to back or back to front

Projected Tetrahedra – Results



A Novel Approach to Visualizing Dark Matter Simulations; Ralf Kaehler, et al. 2012, (link)

Volume Statistics

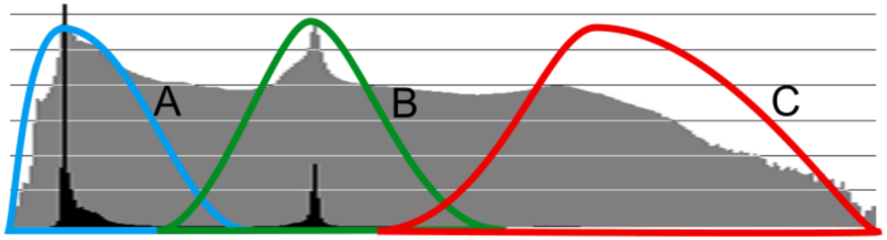
HISTOGRAMS AND SCATTER PLOTS

Continuous Scatter Plots

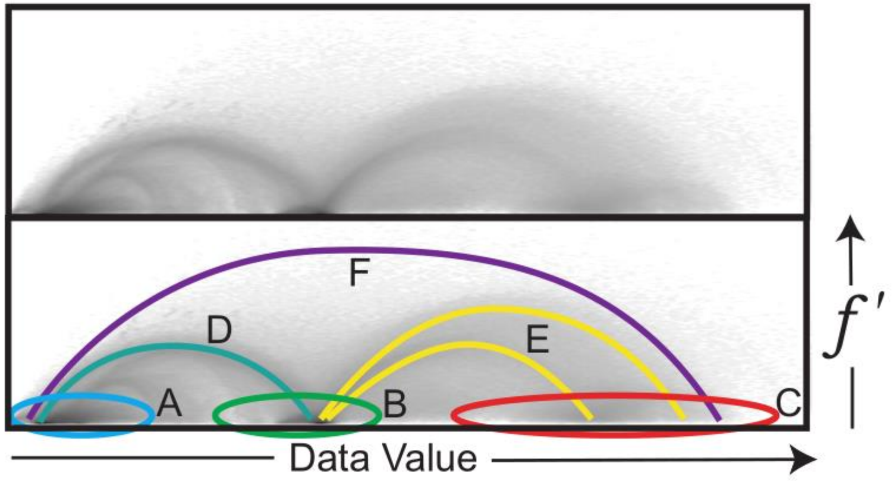
- Histograms / scatter plots are an essential guide for the design of 1D / 2D transfer functions
- Typical features used for second axis of transfer function are
  - gradient magnitude $S' = \|\nabla S\|$
  - Second derivative along gradient direction from Hessian matrix $\boldsymbol{H}$:

$$S'' = \frac{1}{\|\nabla S\|^2} \nabla S^T \boldsymbol{H} \nabla S$$

- discrete computation is based on a binning of the feature values with a bin size parameter per feature



(a) A 1D histogram. The black region represents the number of data value occurrences on a linear scale, the grey is on a log scale. The colored regions (A,B,C) identify basic materials.



(b) A log-scale 2D joint histogram. The lower image shows the location of materials (A,B,C), and material boundaries (D,E,F).

Kniss, J., Kindlmann, G., & Hansen, C. (2002). Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on visualization and computer graphics*, 8(3), 270-285.

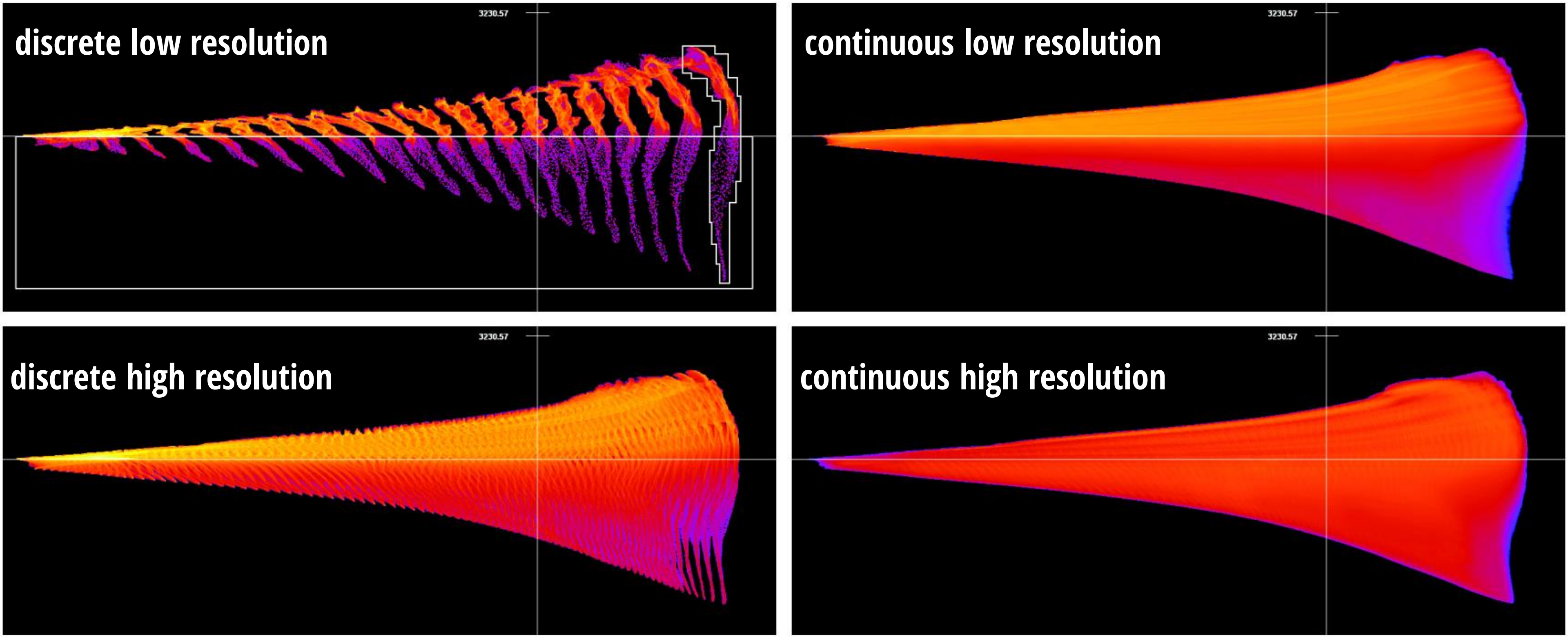
Continuous Scatter Plots

Computer Graphics and Visualization

Bachthaler, S., & Weiskopf, D. (2008). Continuous scatterplots. IEEE transactions on visualization and computer graphics, 14(6), 1428-1435.

Advantages compared to Scatter Plots:

- No overdraw problem
- No missing value problem
- Parameter free



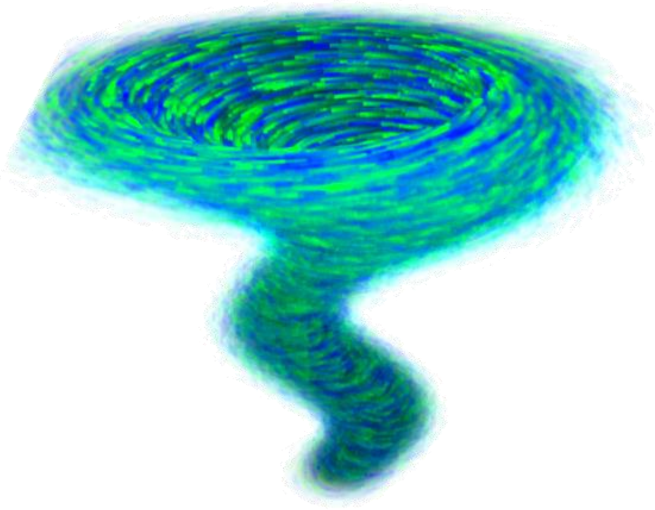
Continuous Scatter Plots

- The interpolation / reconstruction defines a mapping τ from domain points $\underline{x} \in \boldsymbol{R}^n$ to attribute space $\underline{\xi} \in \boldsymbol{R}^m$.

$$\begin{tikzcd}
\boxed{\text{Domain / Observation Space}} & \underline{x} \arrow[r, "\tau"] & \underline{\xi} & \boxed{\text{Attribute / Feature Space}}
\end{tikzcd}$$

$\boldsymbol{R}^n$: independent variables

$\boldsymbol{R}^m$: dependent variables



- In case of 1D attribute space we get a histogram
- In case of 2D attribute space we get a scatter plot
- In case of higher dimensional attribute space one typically works with scatter plot matrices

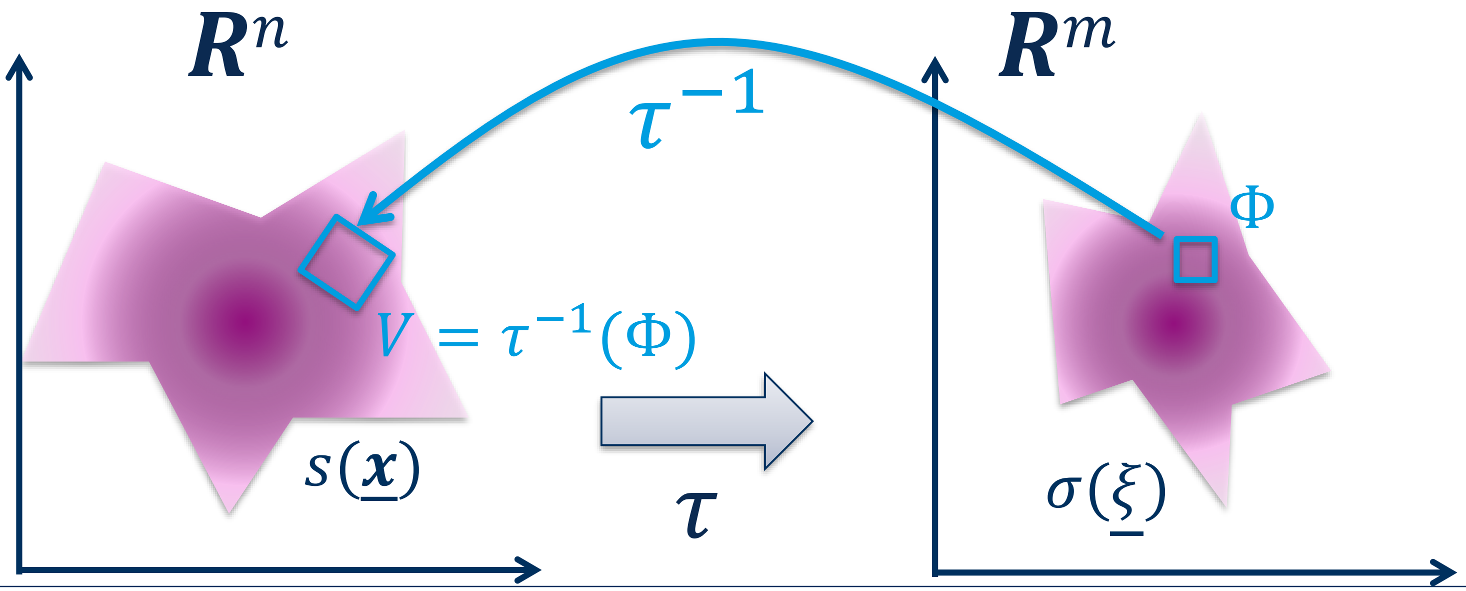
Continuous Scatter Plots

Computer Graphics and Visualization

- Bachthaler et al. introduce the value density $s(\underline{\boldsymbol{x}})$ in observation space and map it to attribute space with τ: via the conservation of the integral over region Φ:

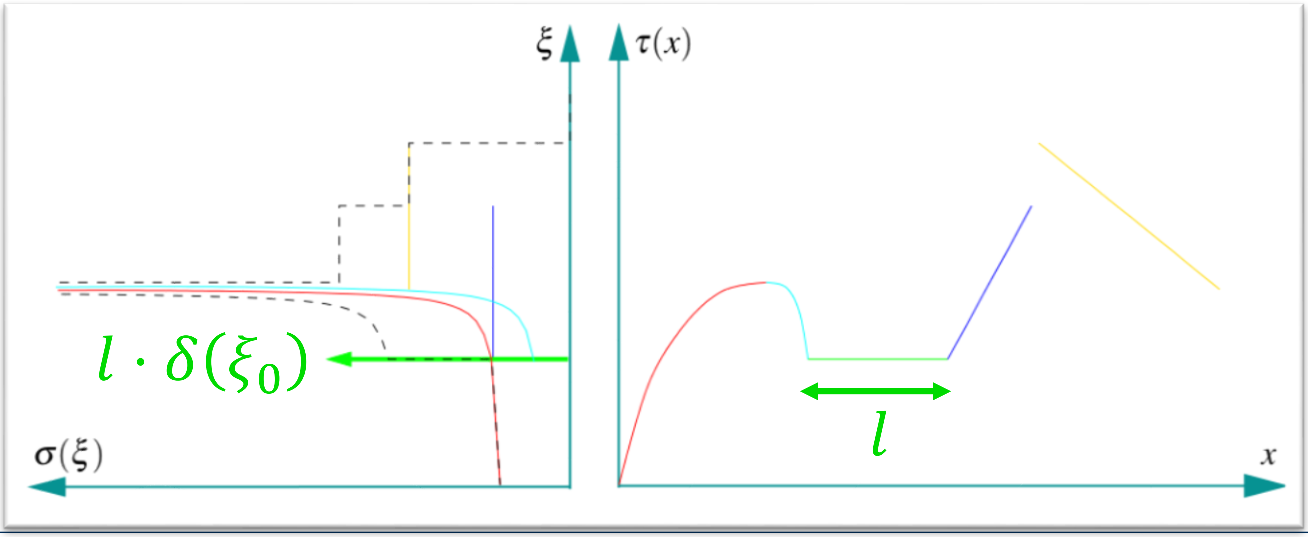
$$\forall\Phi: \int_{\Phi} \sigma\left(\underline{\xi}\right) d\Phi = \int_{V=\tau^{-1}(\Phi)} s(\underline{\boldsymbol{x}})\, dV$$

- Typically $s(\underline{\boldsymbol{x}}) \equiv 1$.



Continuous Scatter Plots

- $\tau^{-1}\left(\underline{\xi}\right)$ maps to the set of pre-images of $\underline{\xi}$.
- For n-dimensional regions V in domain that map to a constant $\underline{\xi}_0$, $\sigma\left(\underline{\xi}_0\right)$ becomes a delta function multiplied by V such that local integration around $\underline{\xi}_0$ yields V
- Illustration for $n = m = 1$ with colored parts summed to dashed continuous histogram:



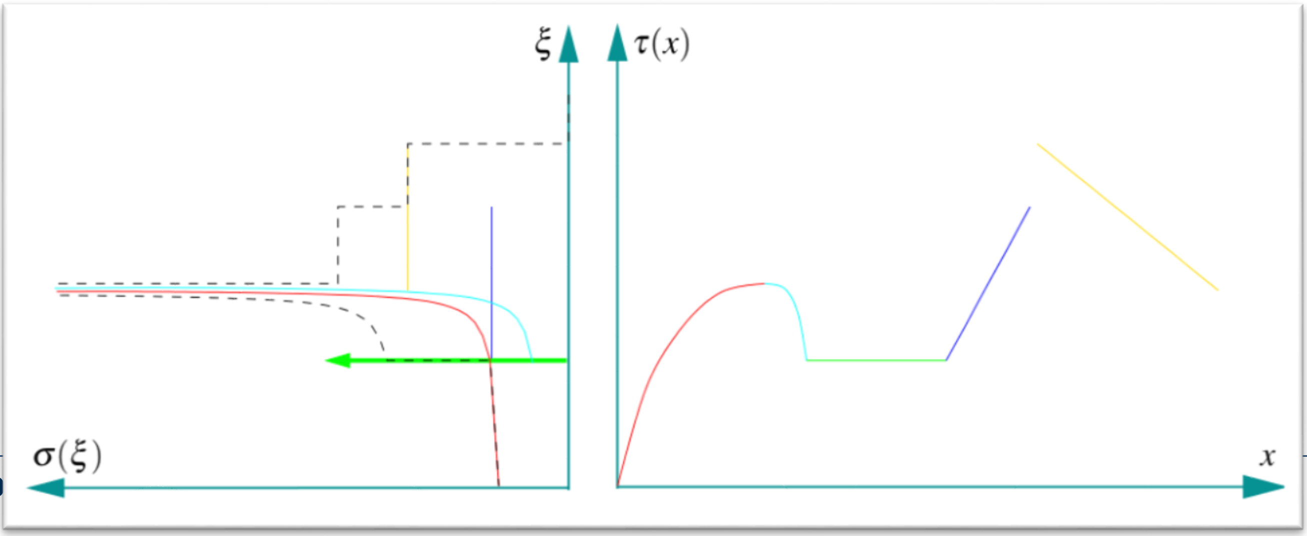
Continuous Scatter Plots – $n = m = 1$

Computer Graphics and Visualization

- For constant slope of $\tau(x) = c \cdot x$, the contribution to the histogram is $d\sigma(\xi) = \frac{1}{|c|}$
- On intervals $I = [x_0, x_1]$ of invertible $\tau|_I(x)$ we get

$$\sigma\Big|_{\tau(I)}(\xi) = \frac{s(x)}{|\partial_x \tau|_I(x)|}$$

- This follows from change of variables in the integral:
$\int_{\tau(x_0)}^{\tau(x_1)} \sigma(\xi) d\xi = \int_{x_0}^{x_1} \sigma\big(\tau(x)\big)|\tau'(x)|dx = \int_{x_0}^{x_1} s(x)dx$



Continuous Scatter Plots – $n = m$

- For $n = m > 1$ this generalizes to regions V of invertible $\tau|_V(x)$ with the Jacobian matrix $\boldsymbol{J}_{\underline{\boldsymbol{x}}}\tau$ of first derivatives to

$$\sigma\Big|_{\tau(V)}\left(\underline{\xi}\right) = \frac{s\left(\underline{\boldsymbol{x}}\right)}{\left|\det\left(\boldsymbol{J}_{\underline{\boldsymbol{x}}}\tau|_V\left(\underline{\boldsymbol{x}}\right)\right)\right|}$$

- Typically we split the domain into cells V_i of invertible regions of τ (e.g. tetrahedral) and compute σ by summing over the cells:

$$\sigma\left(\underline{\xi}\right) = \sum_i \sigma\Big|_{\tau(V_i)}\left(\underline{\xi}\right)$$

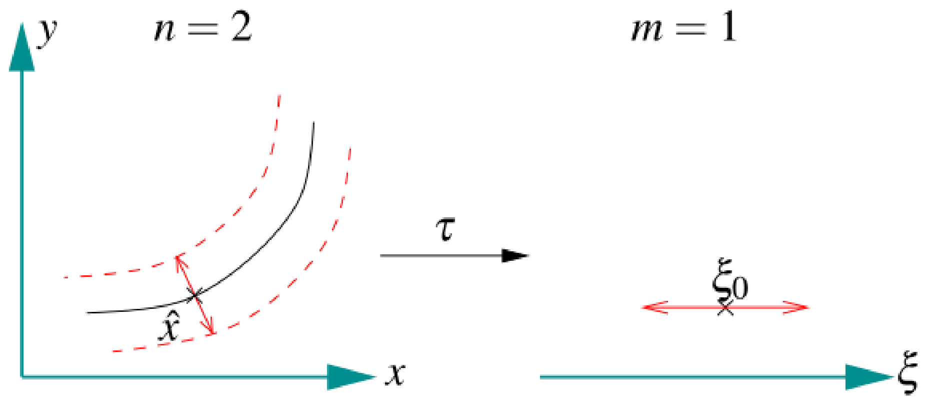
Continuous Scatter Plots – $n > m$

Computer Graphics and Visualization

- The case $n = 2/3$ and $m = 1$ is used to compute a histogram over iso-lines/iso-surfaces
- To compute $\sigma(\xi)$ one has to integrate over the 1D/2D iso-line/iso-surface:

$$\sigma(\xi) = \int_{\tau^{-1}(\xi)} \frac{s(\underline{\boldsymbol{x}})}{\left|\det\left(\boldsymbol{J}_{\underline{\boldsymbol{x}}}\tau(\underline{\boldsymbol{x}})\right)\right|} d^{(n-m)}\underline{\boldsymbol{x}}$$

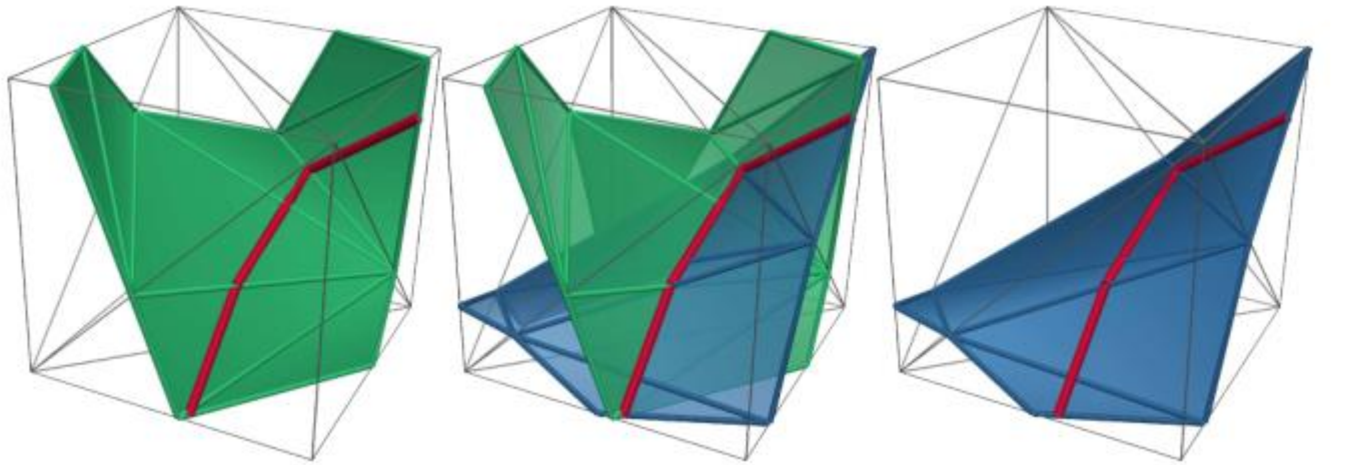
again the determinant of the Jacobian needs to be considerer – not only the length/area of iso-line/surface



Fiber Surfaces

Carr, H., Geng, Z., Tierny, J., Chattopadhyay, A., & Knoll, A. (2015, June). Fiber surfaces: Generalizing isosurfaces to bivariate data. In Computer Graphics Forum (Vol. 34, No. 3, pp. 241-250).

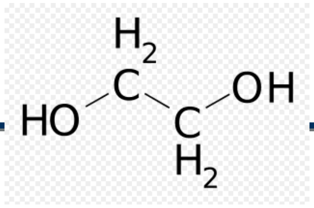
- The case $n = 3$ and $m = 2$ is used to compute a scatter plot over two features
- Each point $\underline{\xi}$ corresponds to a fiber curve resulting from the intersection of two iso-surfaces



ξ_1 iso-surface and fiber **both iso-surface and fiber** **ξ_2 iso-surface and fiber**

- The boundary of a region in the scatter plot corresponds to a fiber family forming a fiber surface

Fiber Surfaces



Computer Graphics and Visualization

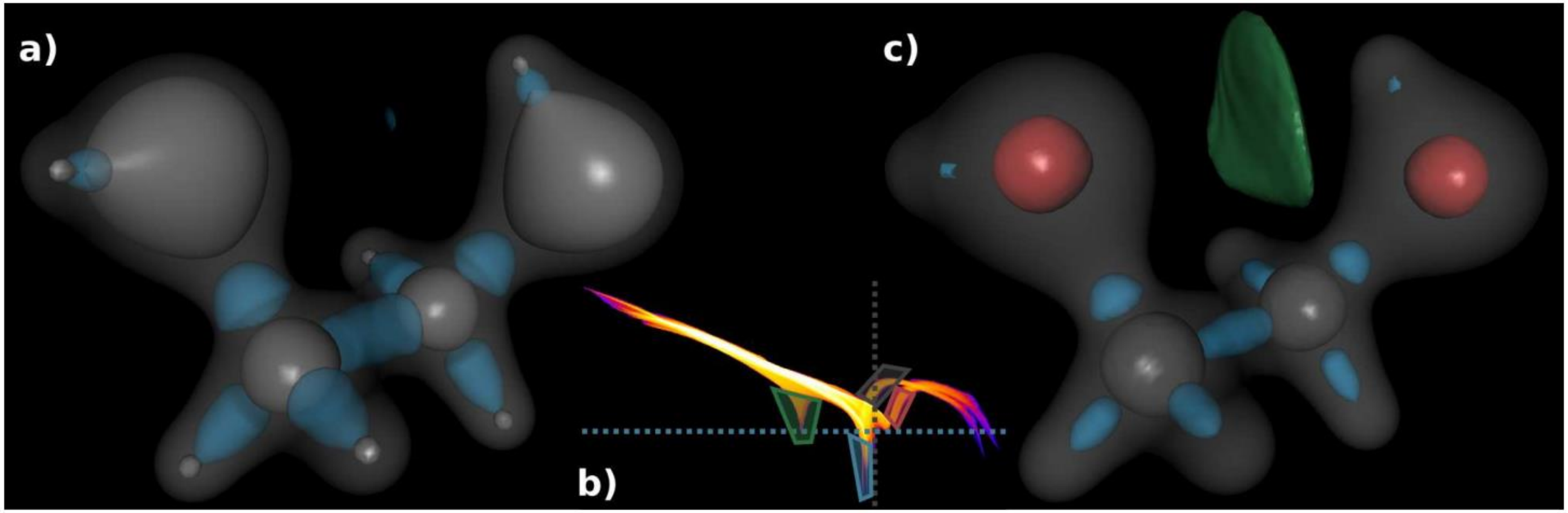


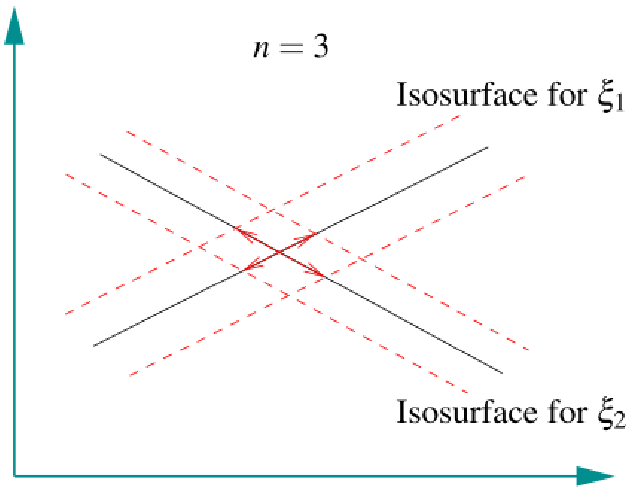
Figure 1: *Fiber Surfaces of electron density and reduced gradient in an ethane-diol molecule: (a) While an isosurface of electron density identifies regions of influence of atoms (grey), it does not distinguish atomic type. An isosurface of reduced gradient identifies bonding interaction sites (blue) but does not distinguish non-covalent (top) from covalent bonds (others). (b) Continuous scatter plot (log scale) of electron density and reduced gradient. Isosurfaces and fiber surfaces are shown as dashed lines and polygons respectively. (c) Fiber surfaces distinguish atom types (oxygen in red, carbons in grey) as well as bond types (non-covalent in green, covalent in blue).*

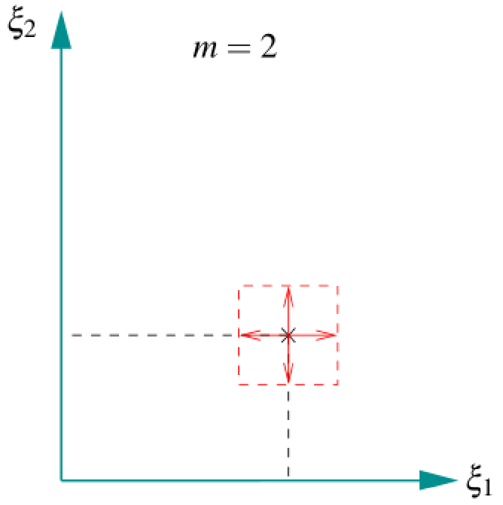
Fiber Surface Scatter Plots

- The case $n = 3$ and $m = 2$ we get a 1D integration along fibers to compute σ:

$$\sigma\left(\underline{\boldsymbol{\xi}} = \begin{pmatrix}\xi_1\\ \xi_2\end{pmatrix} \in \boldsymbol{R}^2\right) = \int_{\tau^{-1}(\underline{\xi})} \frac{s(\underline{\boldsymbol{x}})}{\left|\det\left(\boldsymbol{J}_{\underline{\boldsymbol{x}}}\tau(\underline{\boldsymbol{x}})\right)\right|} d^{(1)}\underline{\boldsymbol{x}}$$

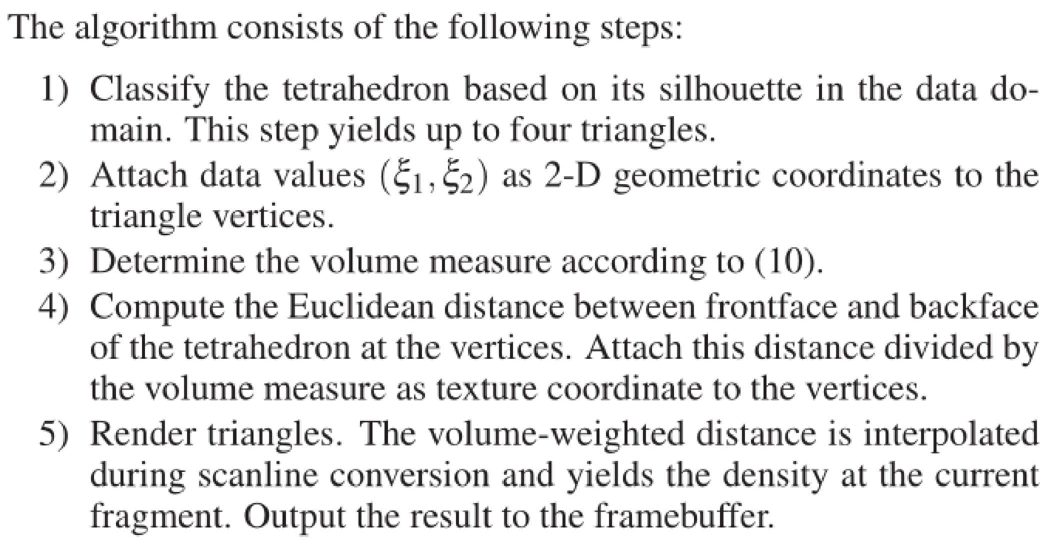
- Where $\det\left(\boldsymbol{J}_{\underline{\boldsymbol{x}}}\tau(\underline{\boldsymbol{x}})\right)$ can be computed from the gradient vectors of $\underline{\boldsymbol{\xi}}$-components: $\det\left(\boldsymbol{J}_{\underline{\boldsymbol{x}}}\tau(\underline{\boldsymbol{x}})\right) = \left\|\nabla_{\underline{\boldsymbol{x}}}\xi_1 \times \nabla_{\underline{\boldsymbol{x}}}\xi_2\right\|$





Fiber Surface Scatter Plots

- For volumetric data given on tet meshes the fiber surface scatter plots can be computed similarly to projected tetrahedral approach without visibility sorting but with additive blending:


The algorithm consists of the following steps:

1) Classify the tetrahedron based on its silhouette in the data domain. This step yields up to four triangles.
2) Attach data values (ξ_1, ξ_2) as 2-D geometric coordinates to the triangle vertices.
3) Determine the volume measure according to (10).
4) Compute the Euclidean distance between frontface and backface of the tetrahedron at the vertices. Attach this distance divided by the volume measure as texture coordinate to the vertices.
5) Render triangles. The volume-weighted distance is interpolated during scanline conversion and yields the density at the current fragment. Output the result to the framebuffer.

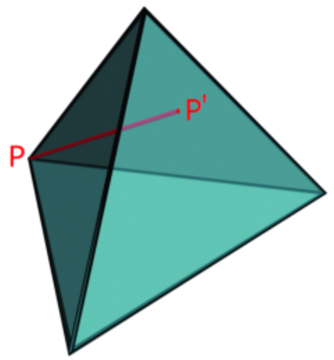


Fig. 7. This illustration shows how the distance is measured that is used to determine the density. At point P, we have to compute a depth value that corresponds to the density. This is done by calculating the distance between P and P' in the spatial domain of the tetrahedron. The point P' is the intersection point between the face opposite to P and the (ξ_1, ξ_2) isoline through P. (Please note that P' does not coincide with a vertex, except for degenerated cases).

Tooth Example + Component Filter

Computer Graphics and Visualization

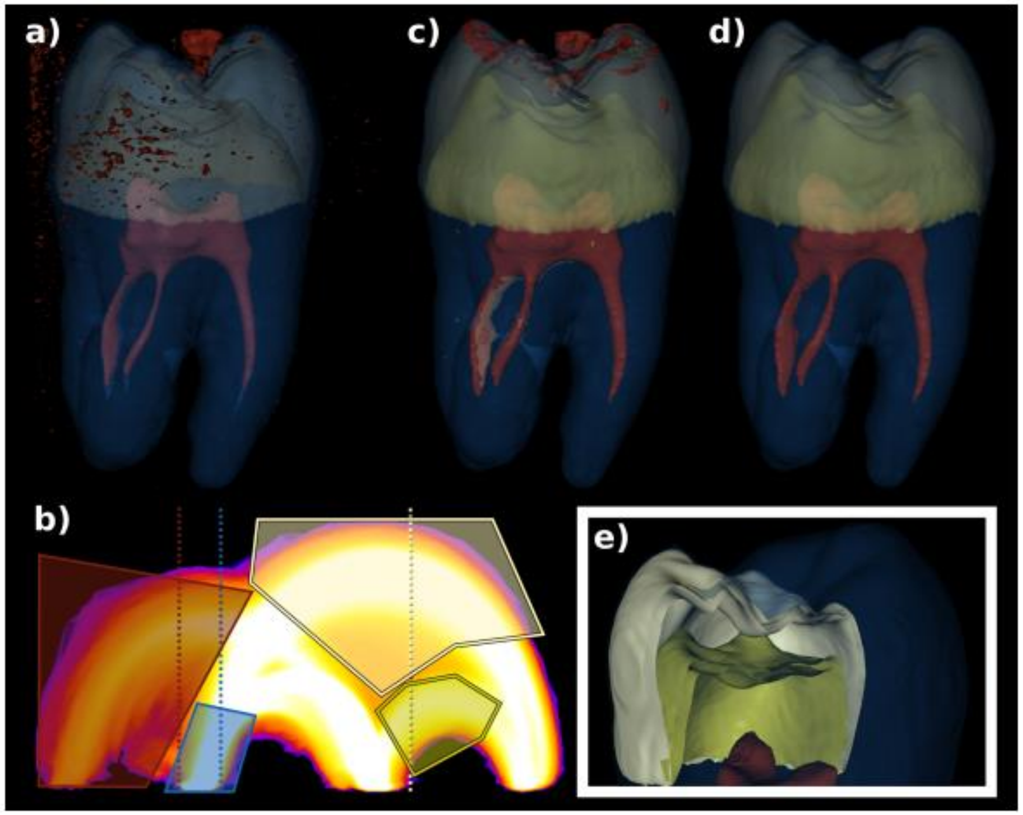


Figure 7: *Fiber surfaces for material boundaries in a tooth CT-scan: (a) User selected isosurfaces. (b) Continuous scatter plot of isovalue vs. gradient magnitude with user selected isovalues (dashed lines) and polygons. (c) Fiber surfaces of the selected polygons. (d) Fiber surfaces after connected component filtering. (e) Cut-away view of the fiber surfaces.*

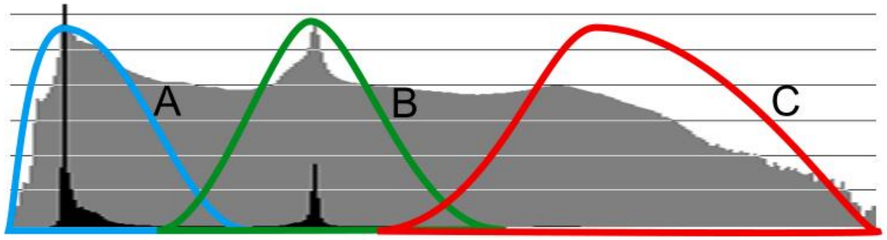
Direct Volume Rendering

MULTI-DIMENSIONAL TRANSFER FUNCTIONS

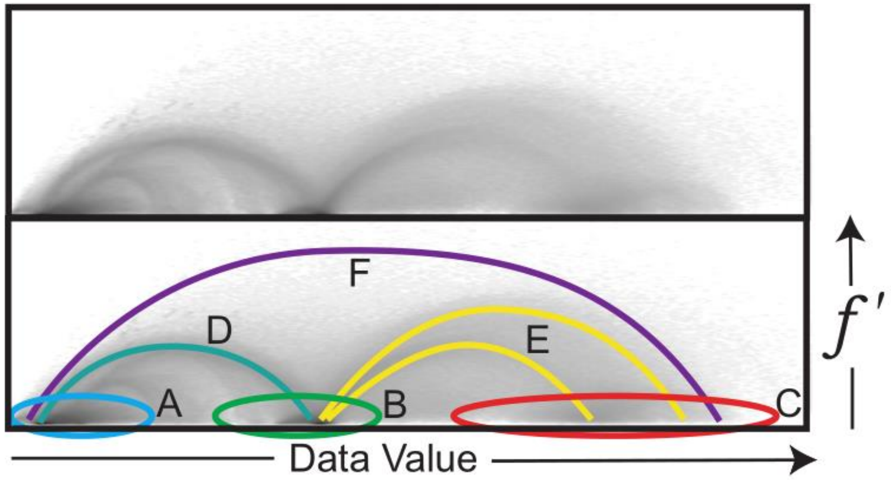
Multi-Dimensional Transfer Functions

Computer Graphics and Visualization

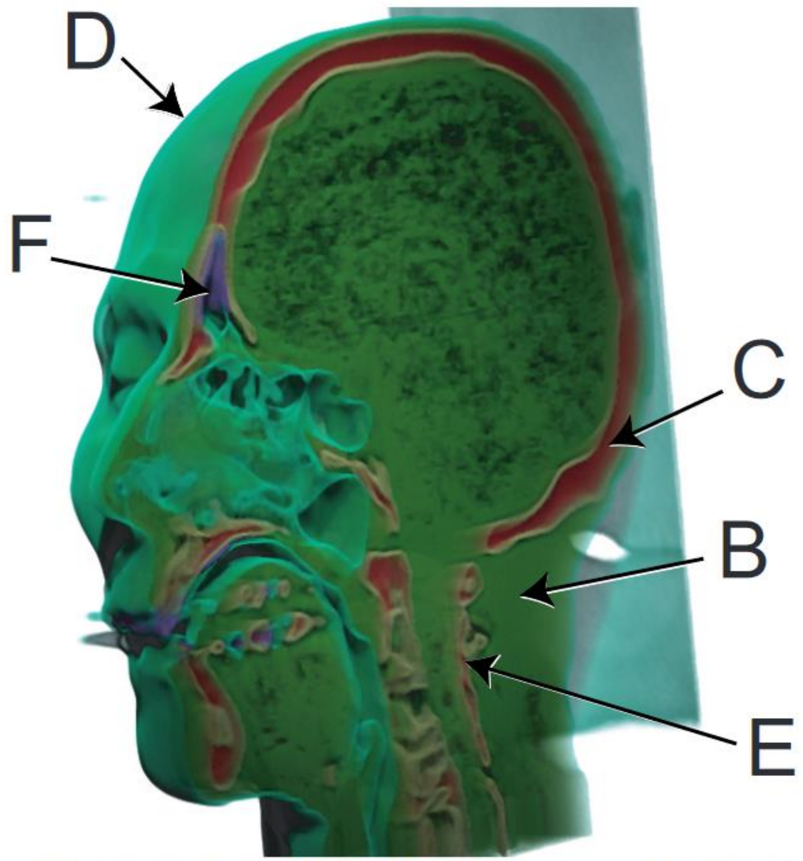
Kniss, J., Kindlmann, G., & Hansen, C. (2002). Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on visualization and computer graphics*, 8(3), 270-285.



(a) A 1D histogram. The black region represents the number of data value occurrences on a linear scale, the grey is on a log scale. The colored regions (A,B,C) identify basic materials. **(air,soft tissue,bone)**



(b) A log-scale 2D joint histogram. The lower image shows the location of materials (A,B,C), and material boundaries (D,E,F).

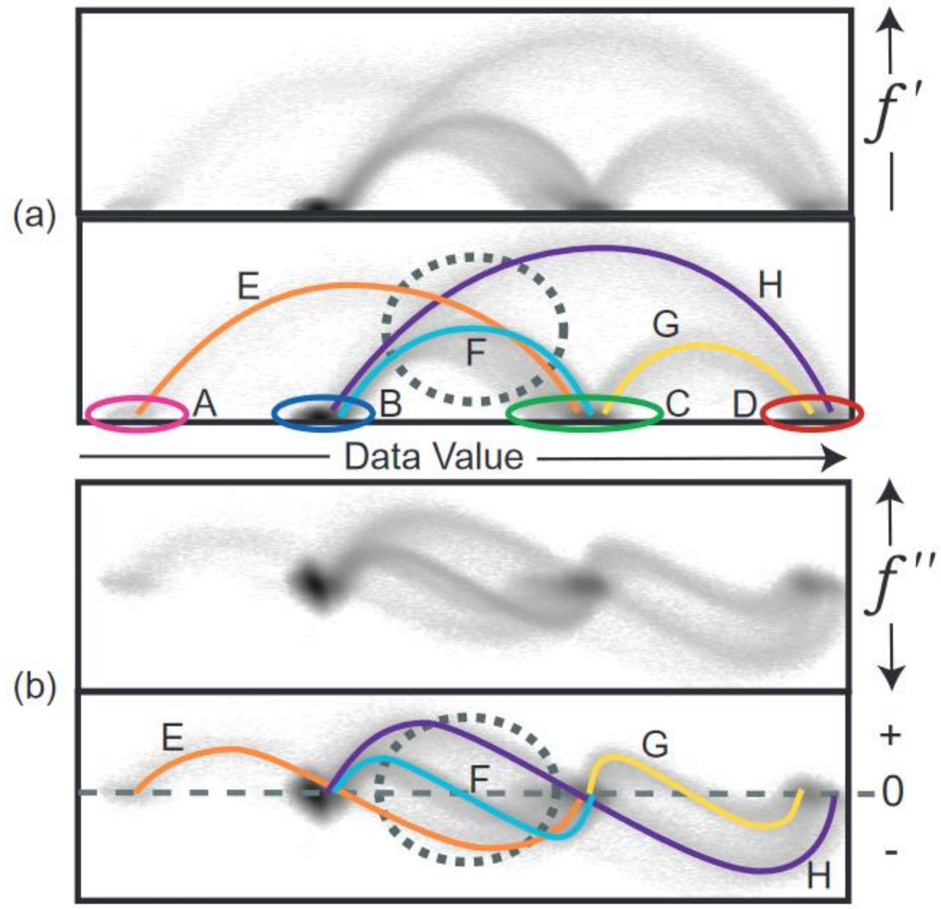


(c) A volume rendering showing all of the materials and boundaries identified above, except air (A), using a 2D transfer function.

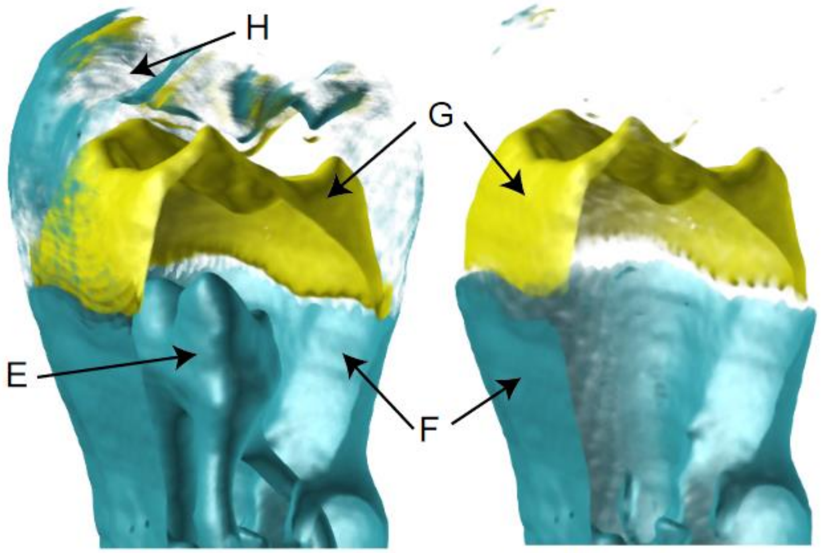
Multi-Dimensional Transfer Functions

Computer Graphics and Visualization

Kniss, J., Kindlmann, G., & Hansen, C. (2002). Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on visualization and computer graphics*, 8(3), 270-285.



(A,B,C,D) = (pulp,background,dentin,enamel)



(c) 2D transfer function (d) 3D transfer function

Figure 3: Material and boundary identification of the human tooth CT with data value and gradient magnitude (f'), seen in (a), and data value and second derivative (f''), seen in (b). The background/dentin boundary (F) cannot be adequately captured with data value and gradient magnitude alone. (c) shows the results of a 2D transfer function designed to show only the background/detin (F) and dentin/enamel boundaries (G). The background/enamel (H) and dentin/pulp (E) boundaries are erroneously colored. Adding the second derivative as a third axis to the transfer function disambiguates the boundaries. (d) shows the results of a 3D transfer function that gives lower opacity to non-zero second derivative values.

Volume

REFERENCES

References

- Keys, R. (1981). Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, *29*(6), 1153-1160.
- Mitchell, D. P., & Netravali, A. N. (1988, August). Reconstruction filters in computer-graphics. In *ACM Siggraph Computer Graphics* (Vol. 22, No. 4, pp. 221-228). ACM.
- Moller, T., Machiraju, R., Mueller, K., & Yagel, R. (1996, October). Classification and local error estimation of interpolation and derivative filters for volume rendering. In *proceedings of 1996 Symposium on Volume Visualization* (pp. 71-78). IEEE.
- Ruijters, D., ter Haar Romeny, B. M., & Suetens, P. (2008). Efficient GPU-based texture interpolation using uniform B-splines. *Journal of Graphics Tools*, *13*(4), 61-69.
- Adams, A., Baek, J., & Davis, M. A. (2010, May). Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum* (Vol. 29, No. 2, pp. 753-762). Oxford, UK: Blackwell Publishing Ltd.
- Yoshizawa, S., Belyaev, A., & Yokota, H. (2010, March). Fast gauss bilateral filtering. In *Computer Graphics Forum* (Vol. 29, No. 1, pp. 60-74). Oxford, UK: Blackwell Publishing Ltd.
- Marks, Joe, et al. "Design galleries: A general approach to setting parameters for computer graphics and animation." Proceedings of the 24th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 1997.

References

- Williams, P. L. (1992). Visibility-ordering meshed polyhedra. ACM Transactions on Graphics (TOG), 11(2), 103-126.
- Max, Hanrahan, Crawfis, Area and volume coherence for efficient visualization of 3D scalar fields, ACM Computer Graphics 24(5), 27-33, 1990
- Kniss, J., Kindlmann, G., & Hansen, C. (2002). Multidimensional transfer functions for interactive volume rendering. IEEE Transactions on visualization and computer graphics, 8(3), 270-285.
- Kruger, J., & Westermann, R. (2003, October). Acceleration techniques for GPU-based volume rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (p. 38). IEEE Computer Society.
- Callahan, S. P., Ikits, M., Comba, J. L. D., & Silva, C. T. (2005). Hardware-assisted visibility sorting for unstructured volume rendering. IEEE Transactions on Visualization and Computer Graphics, 11(3), 285-295.
- Bachthaler, S., & Weiskopf, D. (2008). Continuous scatterplots.
  IEEE transactions on visualization and computer graphics, 14(6), 1428-1435.
- Carr, H., Geng, Z., Tierny, J., Chattopadhyay, A., & Knoll, A. (2015, June). Fiber surfaces: Generalizing isosurfaces to bivariate data. In Computer Graphics Forum (Vol. 34, No. 3, pp. 241-250).