University of Stuttgart
Visualization Research Center (VISUS)

Voronoi-Based Foveated Volume Rendering

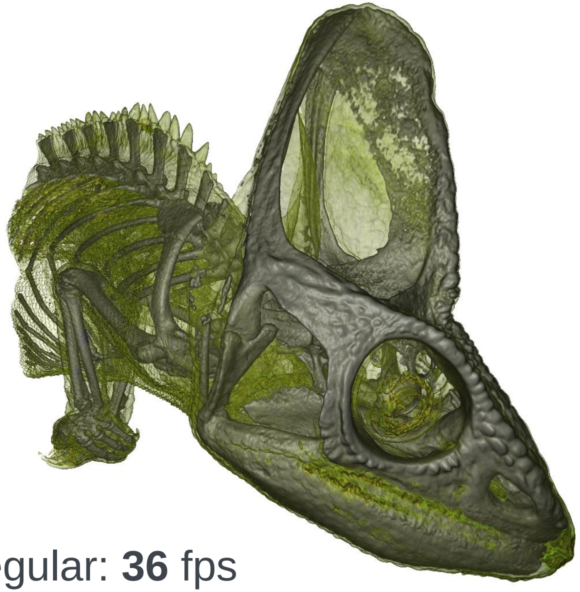
Valentin Bruder, Christoph Schulz, Ruben Bauer, Steffen Frey, Daniel Weiskopf, Thomas Ertl

23-Jul-19

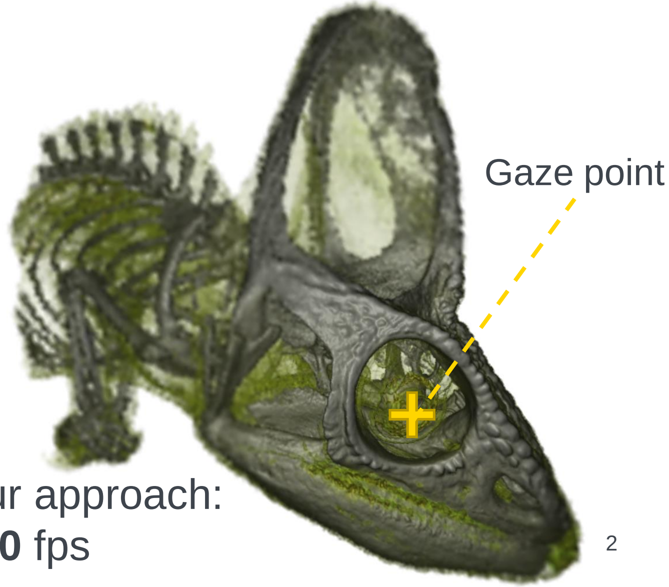
Motivation

- Output devices: increase in pixel density and refresh rate

  → Impacts volume rendering performance
- Typical acceleration techniques based on data properties
- Contributions
  - **Foveated** volume rendering
    - Speedup of factor **1.8 - 3.6**
    - Barely perceptible changes in quality
  - Sampling strategy specific to volume raycasting
  - Voronoi-based sampling and reconstruction

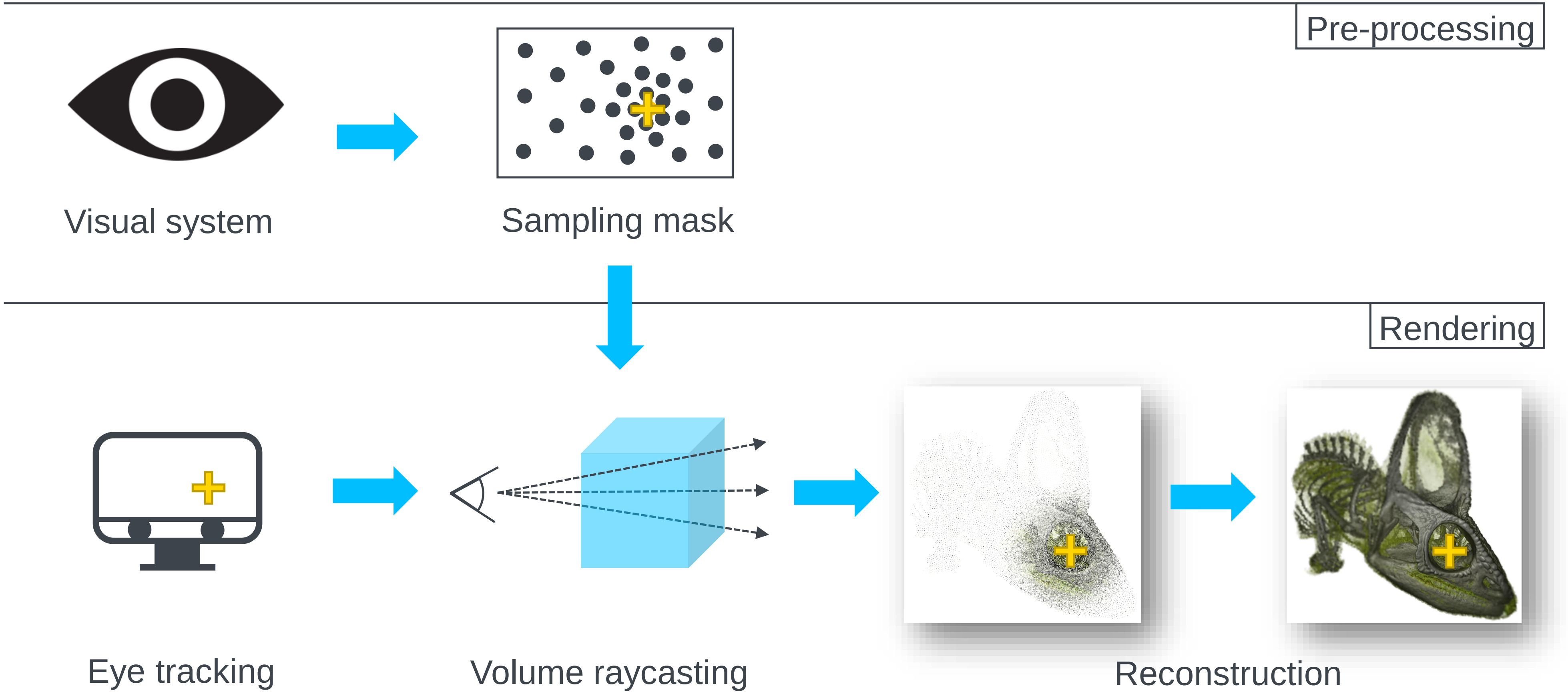


Regular: **36** fps

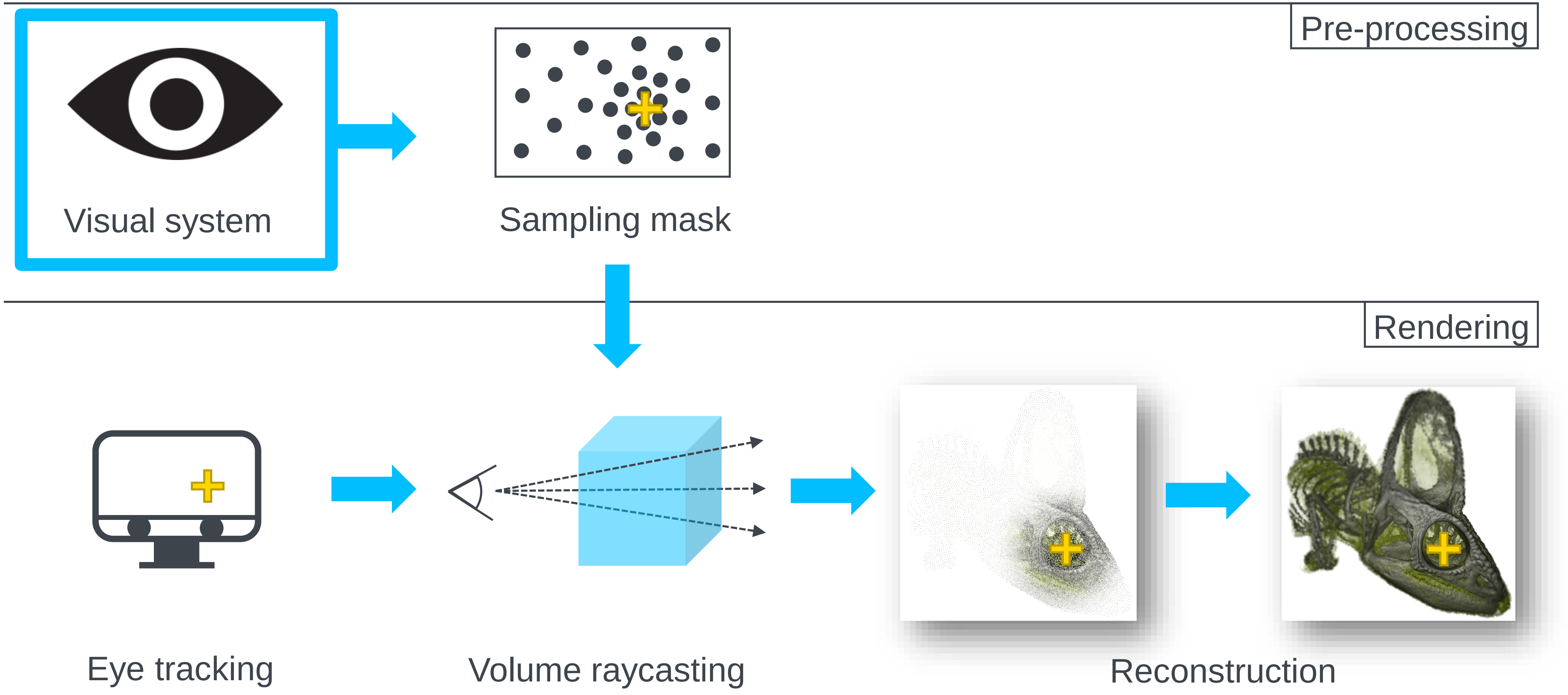


Our approach:
100 fps

Pipeline Overview

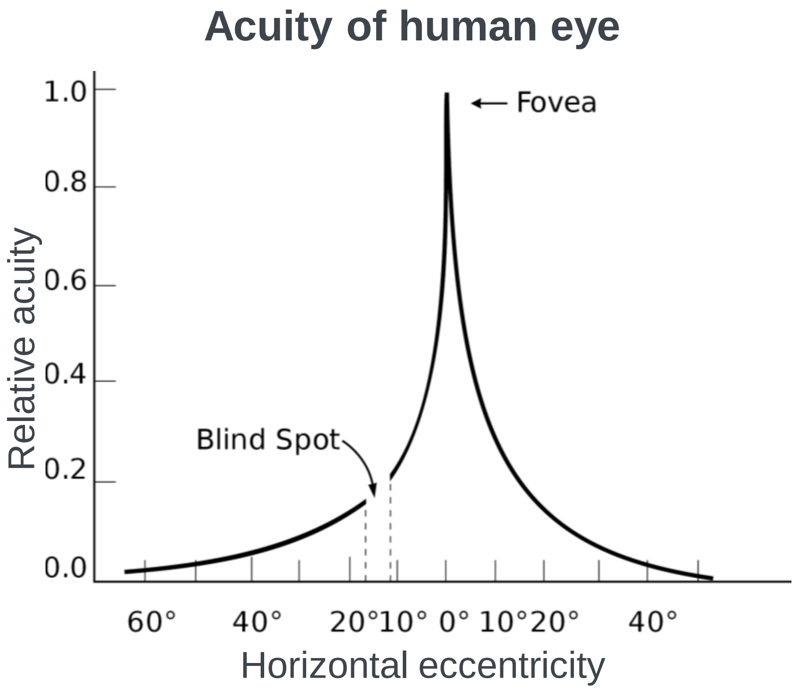


Pipeline Overview



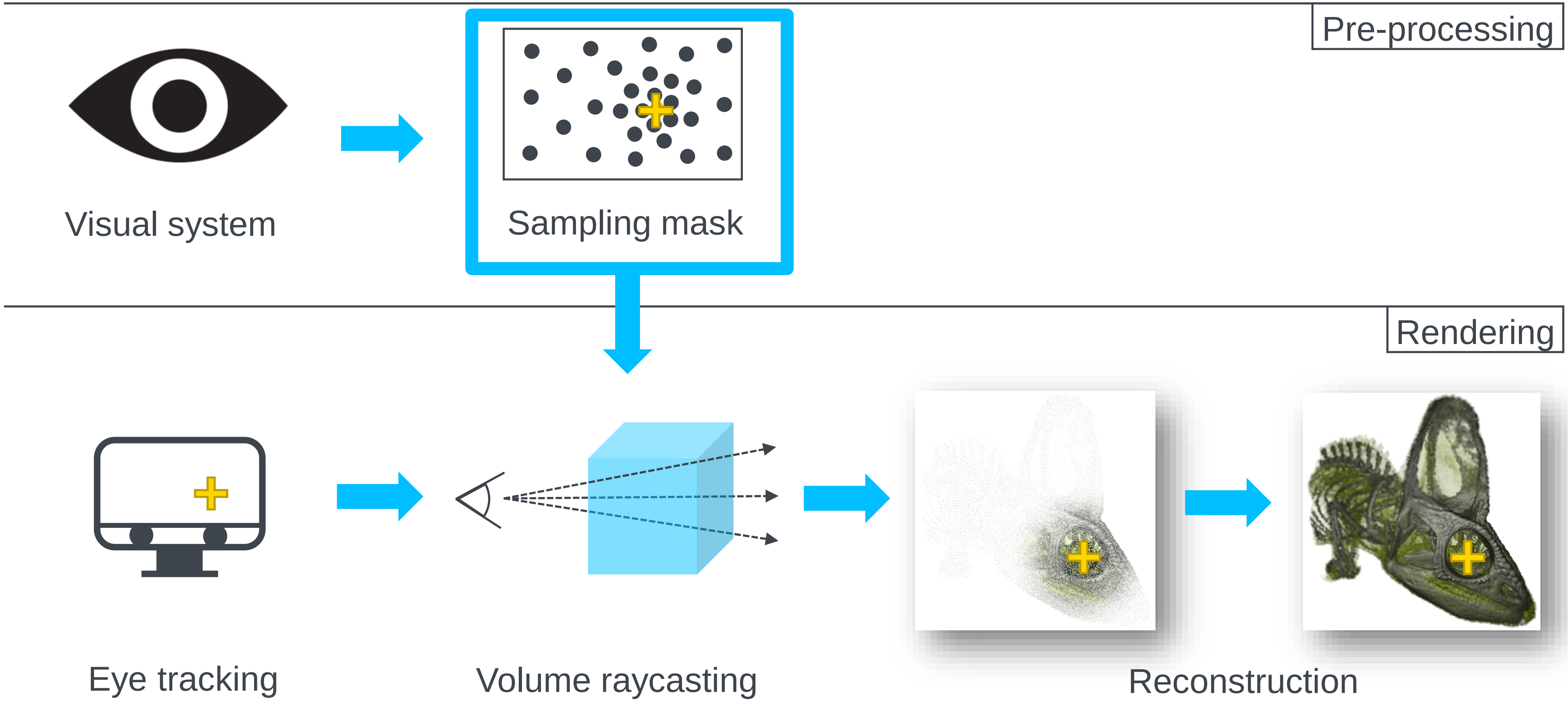
Visual Acuity Fall-Off Model

- Visual acuity falls off in the periphery
- We approximate the fall-off with a 2D Gaussian, using
  - Screen resolution and size
  - Viewing distance
  - Conservative foveal acuity (adults below age 50)



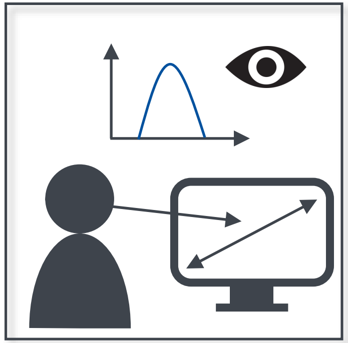
https://commons.wikimedia.org/wiki/File:AcuityHumanEye.svg

Pipeline Overview

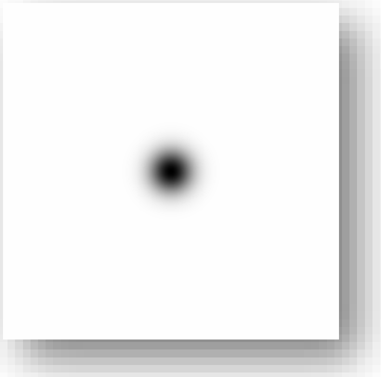


Sampling Mask Generation

Pre-processing



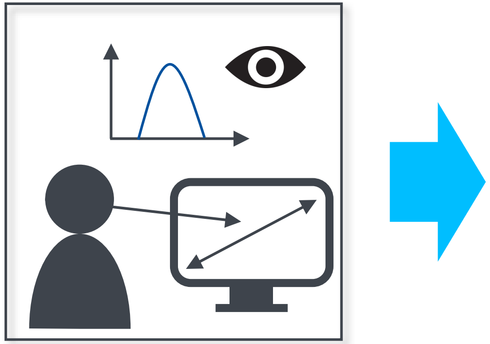
Model visual acuity fall-off



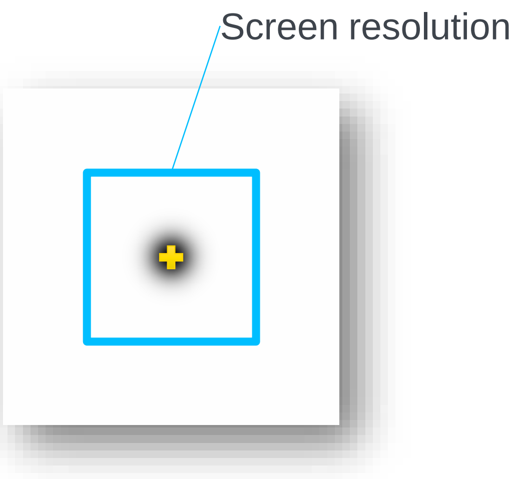
Approximate with Gaussian

Sampling Mask Generation

Pre-processing



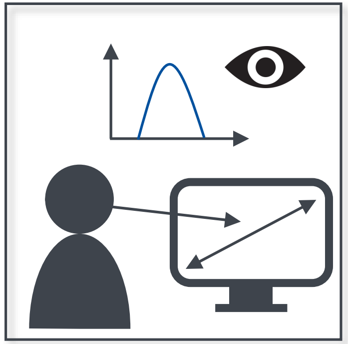
Model visual acuity fall-off

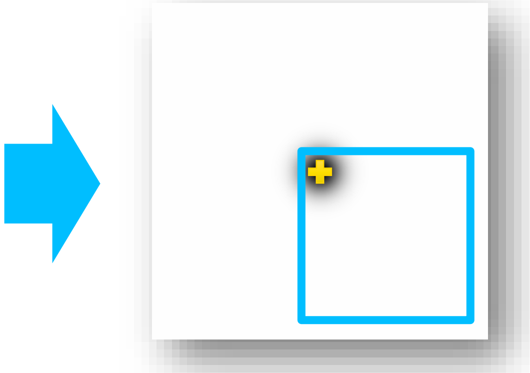


Approximate with Gaussian

Sampling Mask Generation

Pre-processing



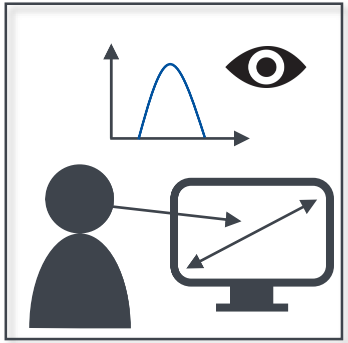


Model visual acuity fall-off

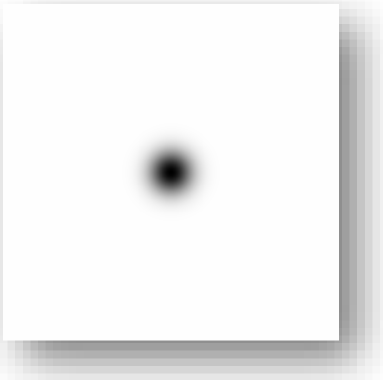
Approximate with Gaussian

Sampling Mask Generation

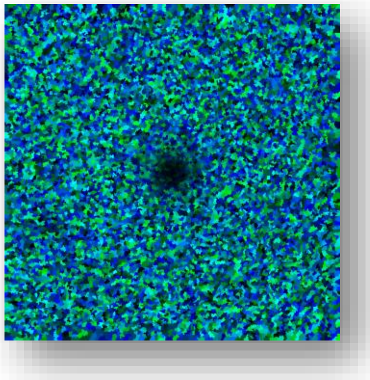
Pre-processing



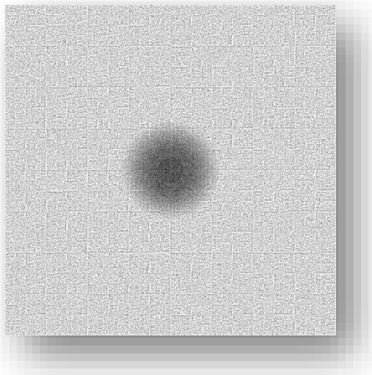
Model visual acuity fall-off



Approximate with Gaussian



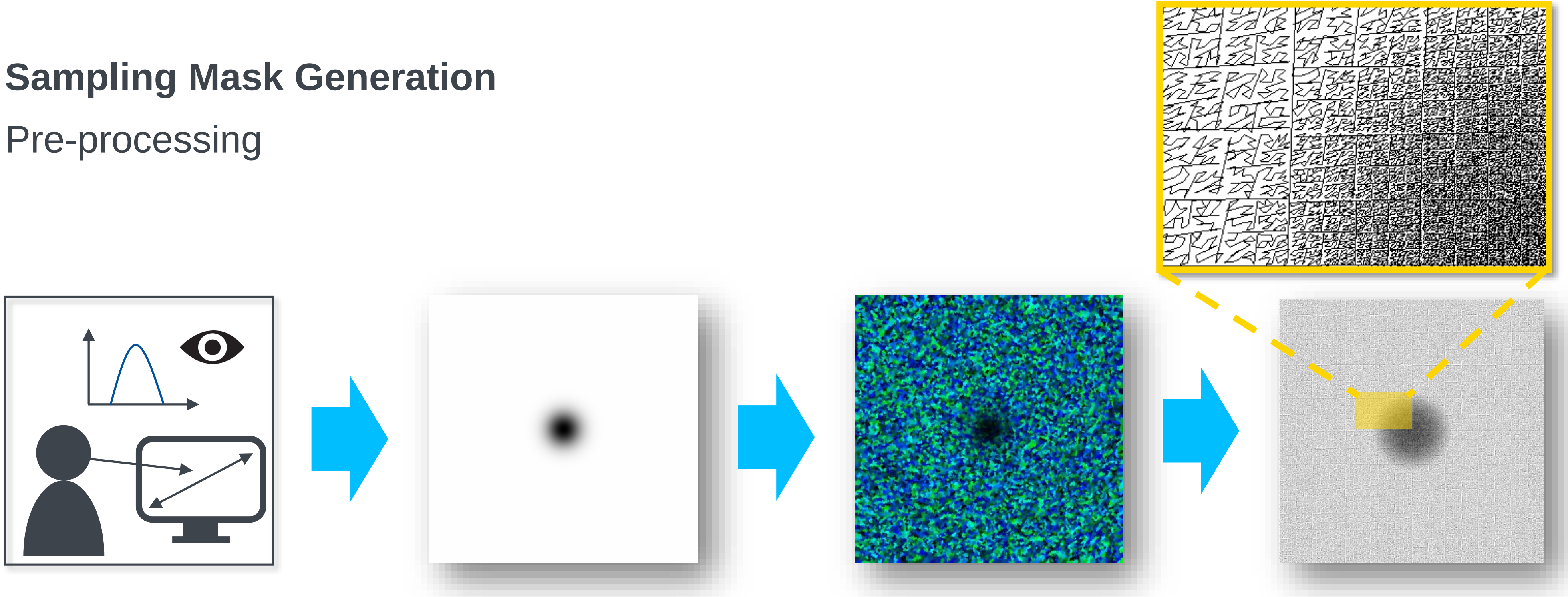
Create sampling mask



Morton ordering (optimization)

Sampling Mask Generation

Pre-processing



Model visual acuity fall-off

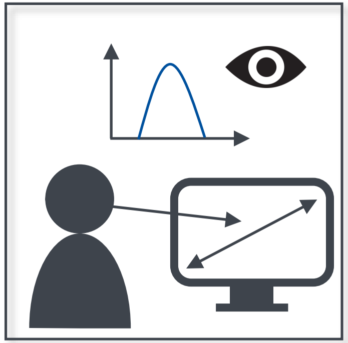
Approximate with Gaussian

Create sampling mask

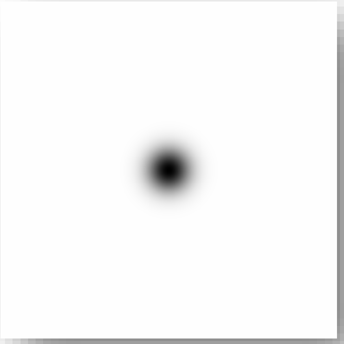
Morton ordering (optimization)

Sampling Mask Generation

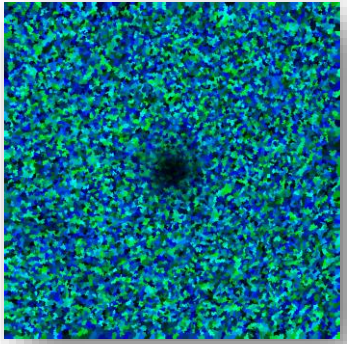
Pre-processing



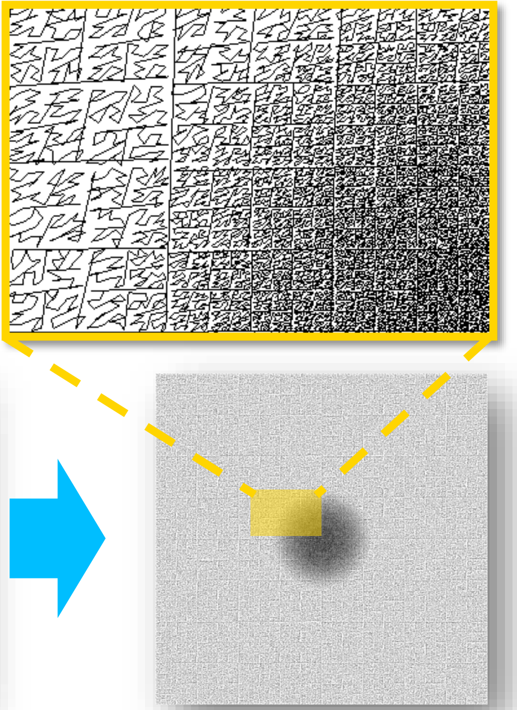
Model visual acuity fall-off



Approximate with Gaussian



Create sampling mask



Morton ordering (optimization)

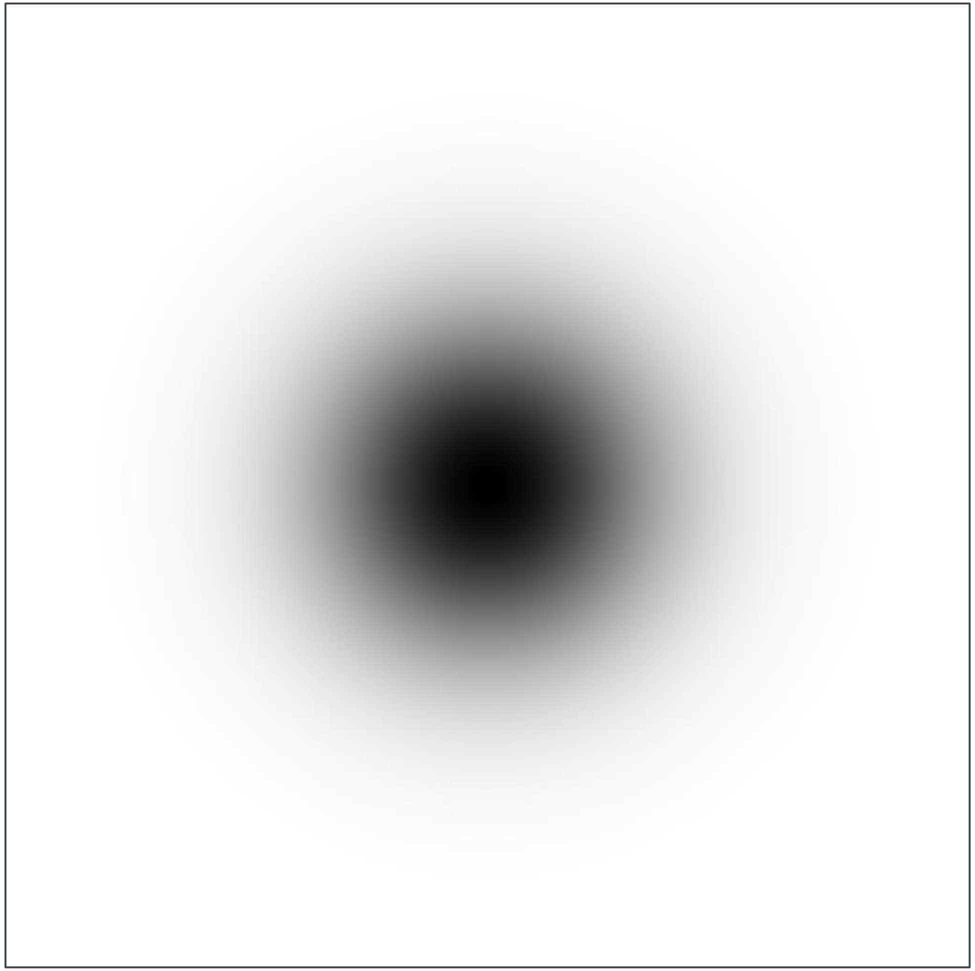
Sampling Mask Generation

Weighted Linde-Buzo-Gray Algorithm

[Deussen, 2017] [Görtler, 2019]

- Arrange samples according to density function
- Little or no visible patterns
- Here: samples → ray starting position
- Voronoi

- One Voronoi cell per sample
- Integrate over each cell: How well is the density function represented?
- Adapt iteratively



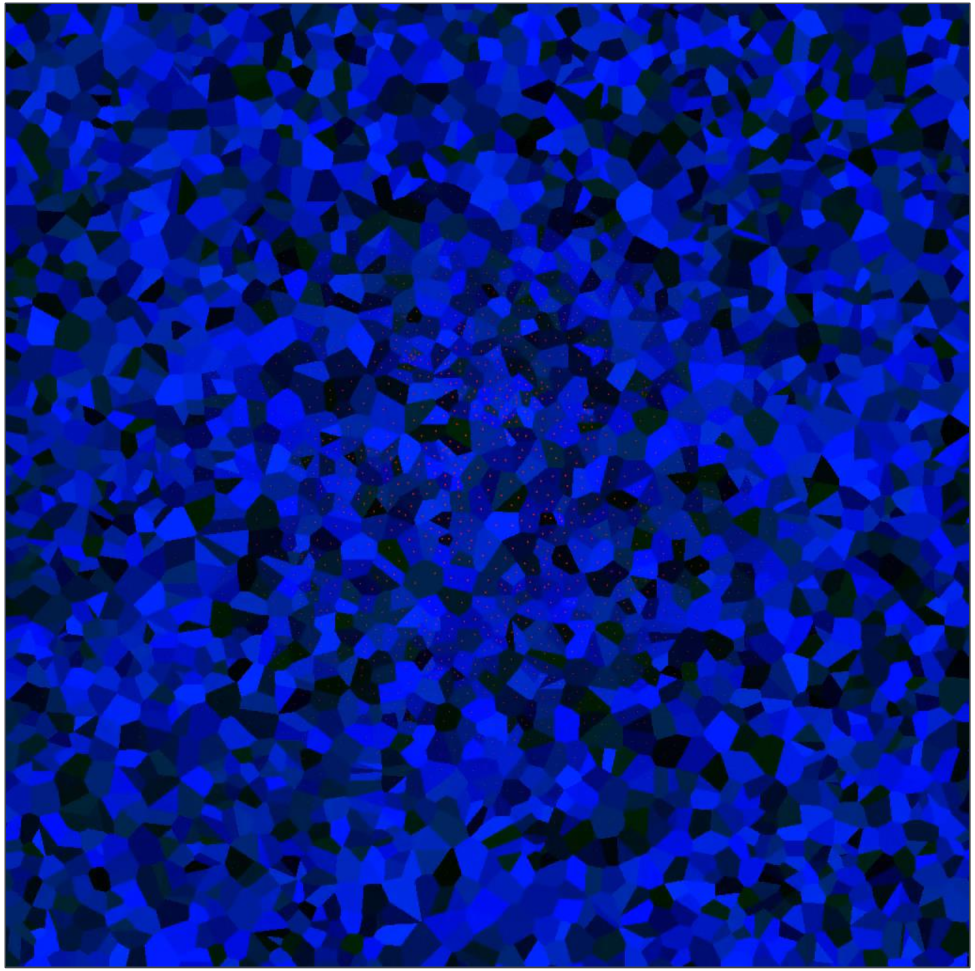
Sampling Mask Generation

Weighted Linde-Buzo-Gray Algorithm

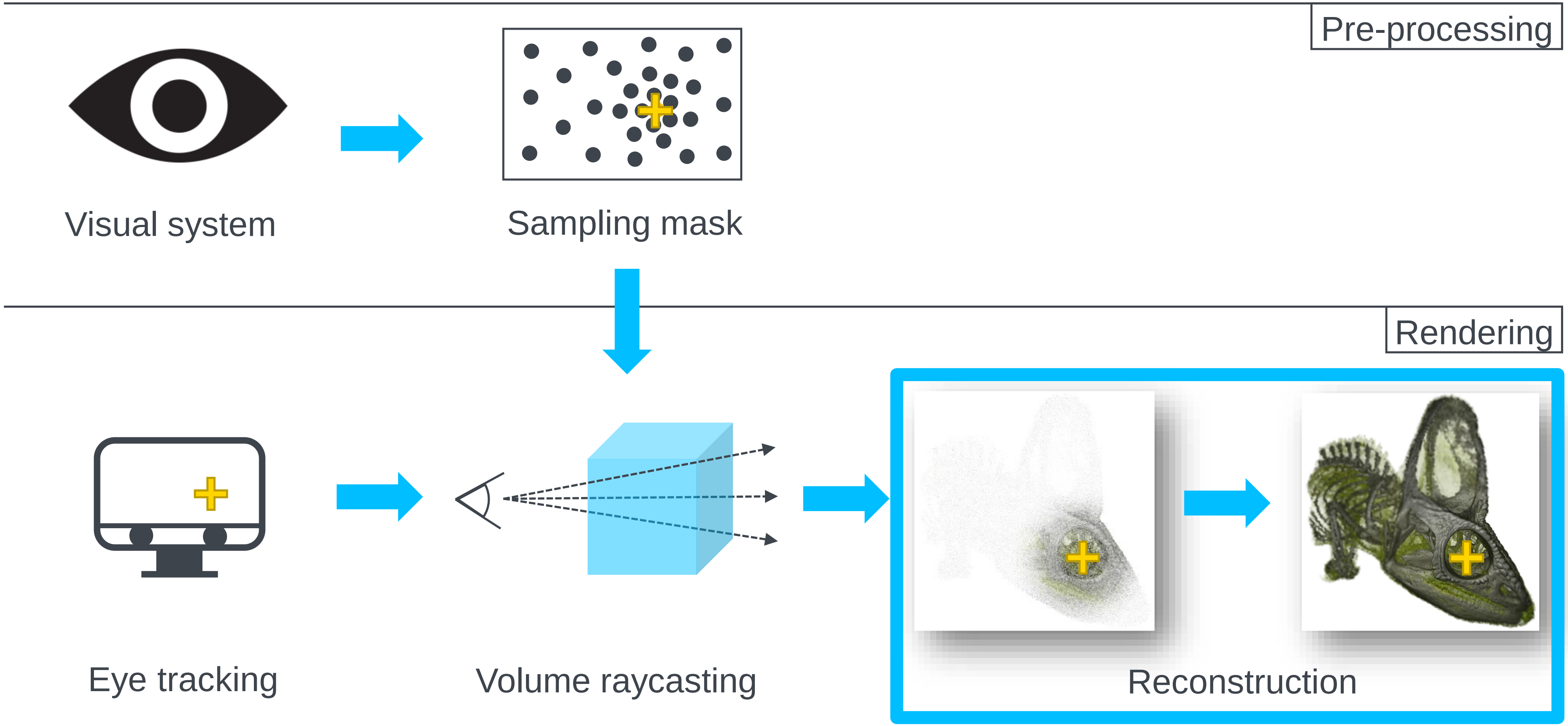
[Deussen, 2017] [Görtler, 2019]

- Arrange samples according to density function
- Little or no visible patterns
- Here: samples → ray starting position
- Voronoi

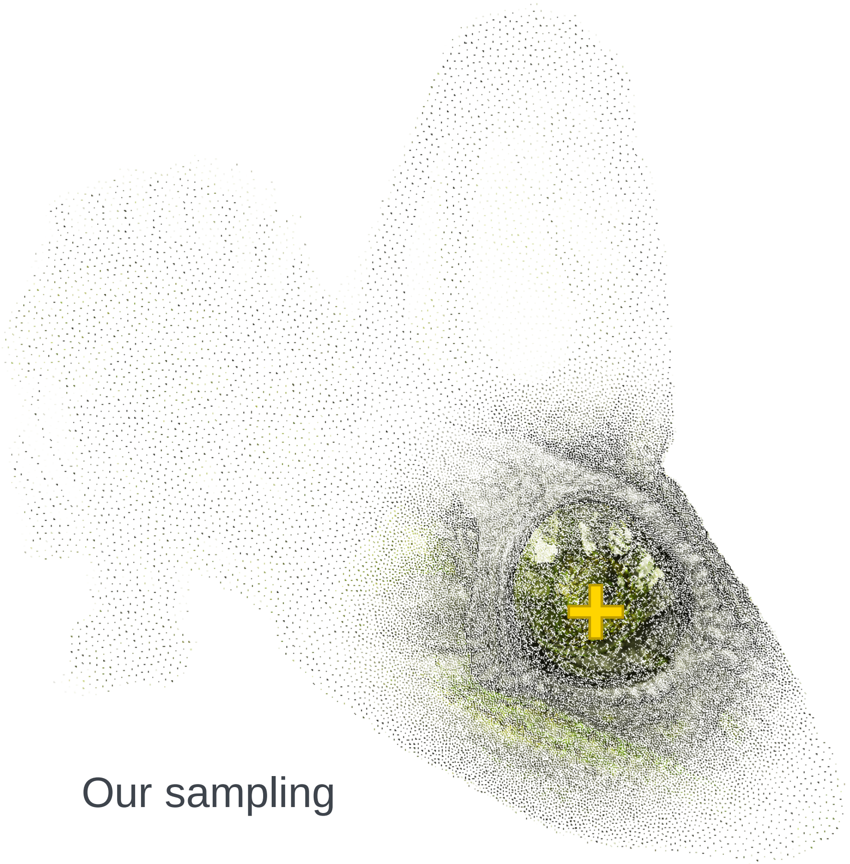
- One Voronoi cell per sample
- Integrate over each cell: How well is the density function represented?
- Adapt iteratively



Pipeline Overview

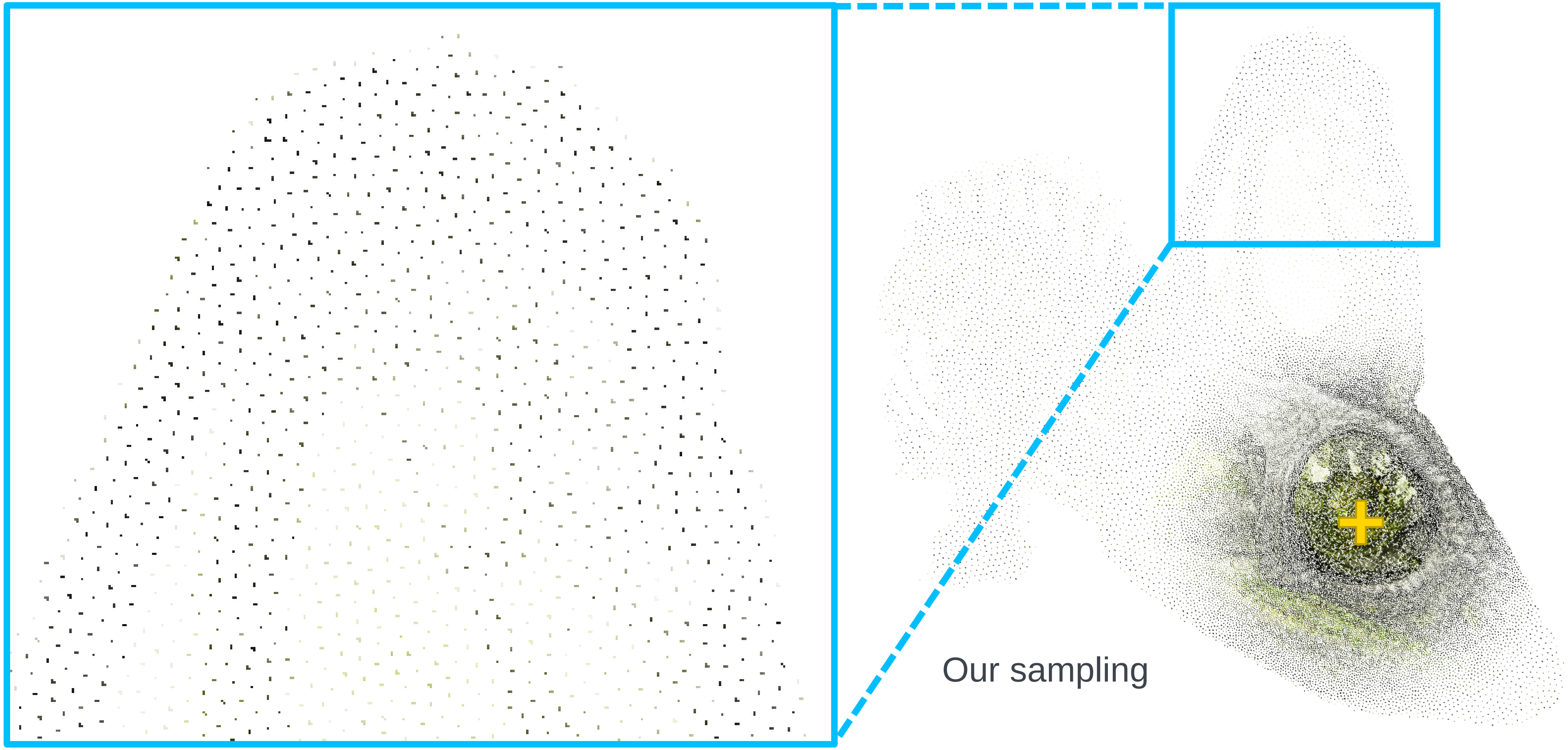


Reconstruction

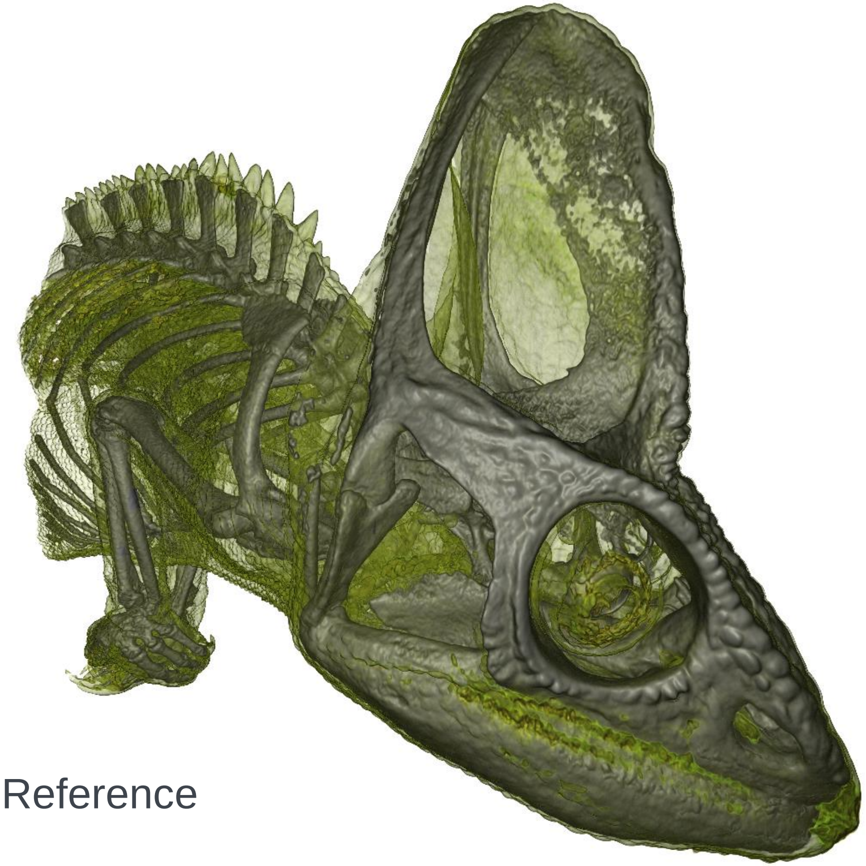


Our sampling

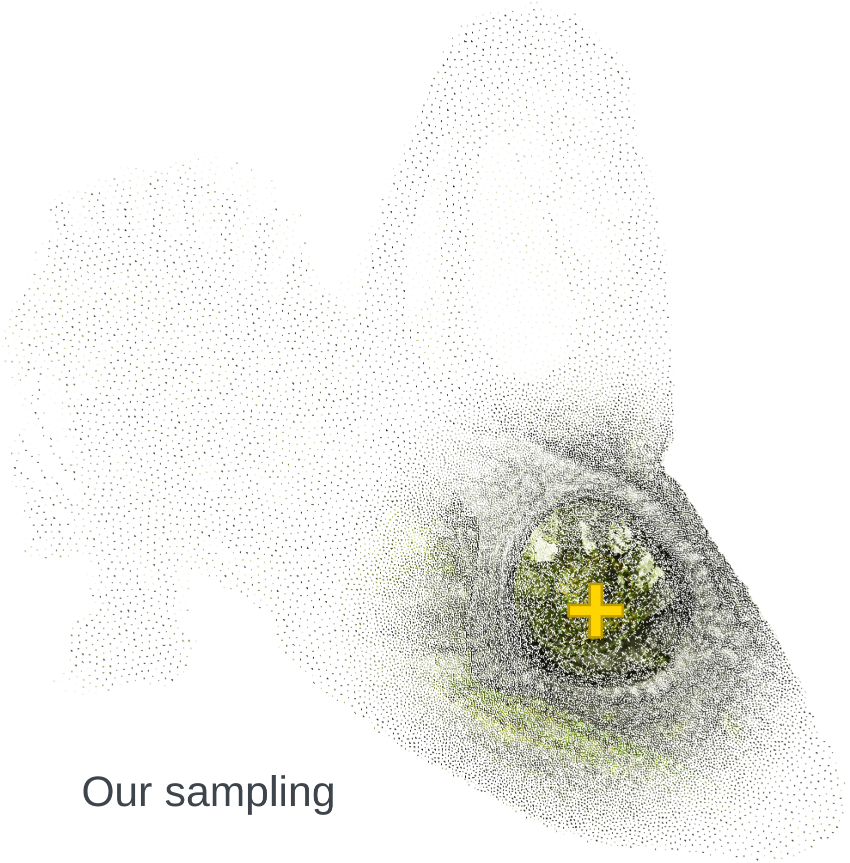
Reconstruction



Reconstruction

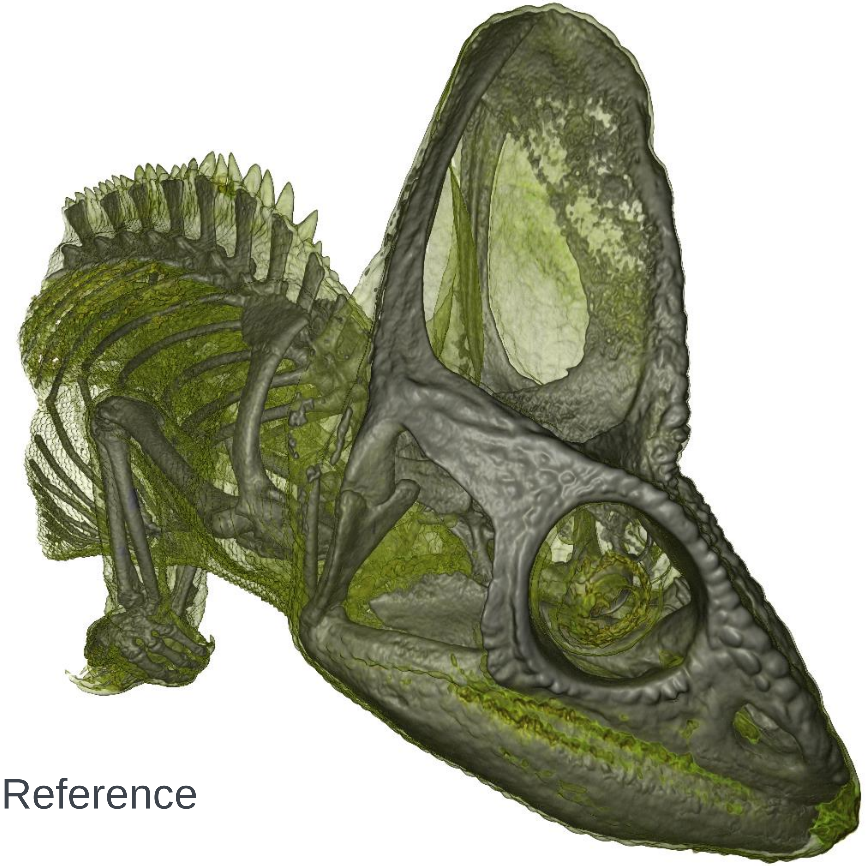


Reference



Our sampling

Reconstruction

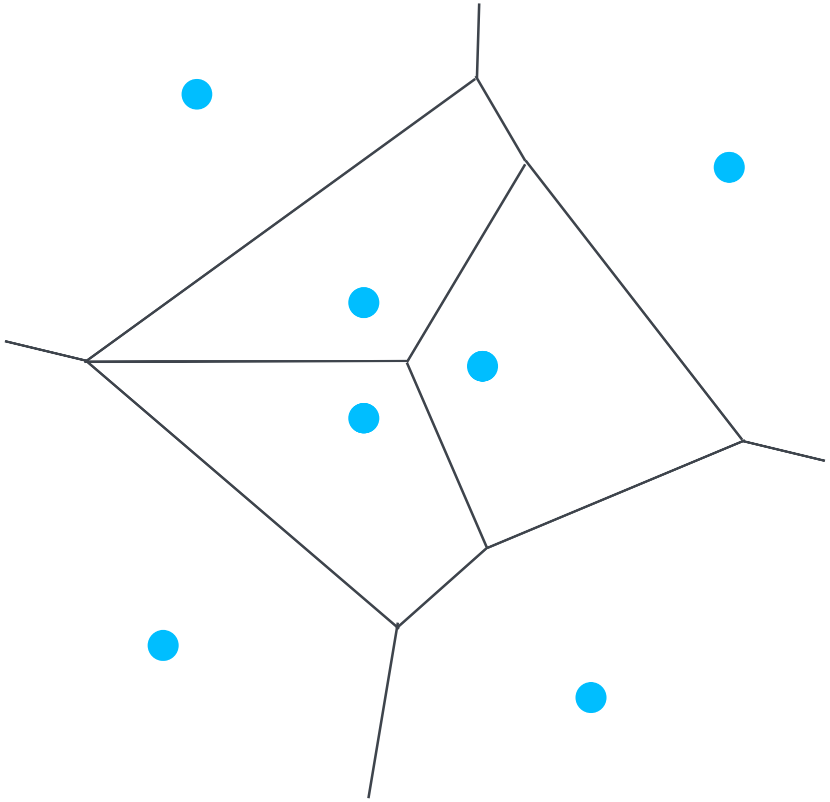


Reference

Our sampling
with reconstruction

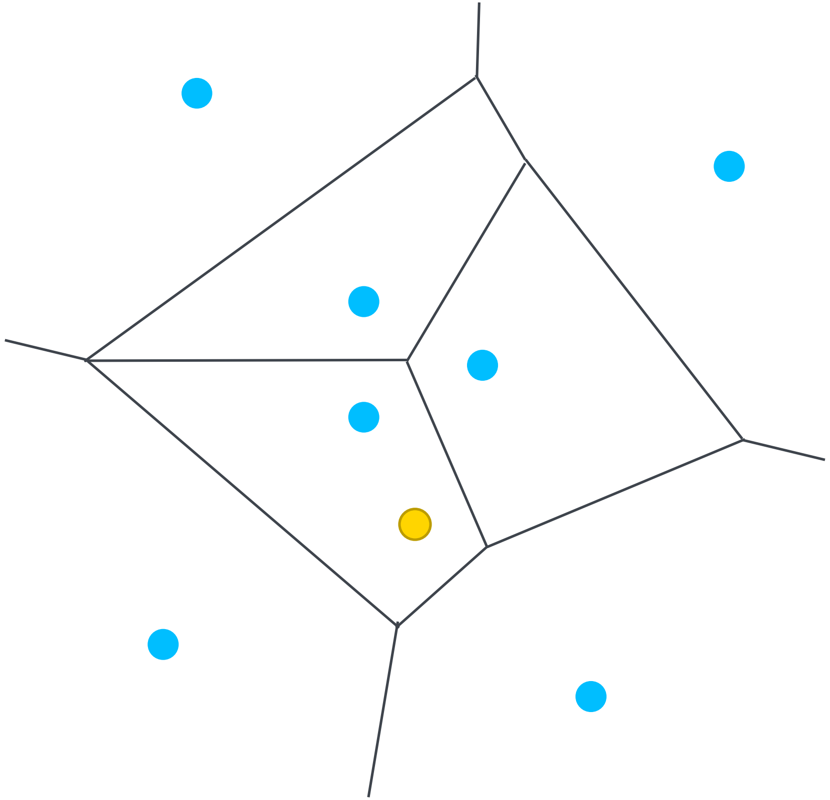
Natural-Neighbor-Based Image Reconstruction

- Natural neighbor interpolation [Sibson, 1980]
  - Smooth approximation
  - Local neighbors
  - Generally C_1 continuous
  - Voronoi



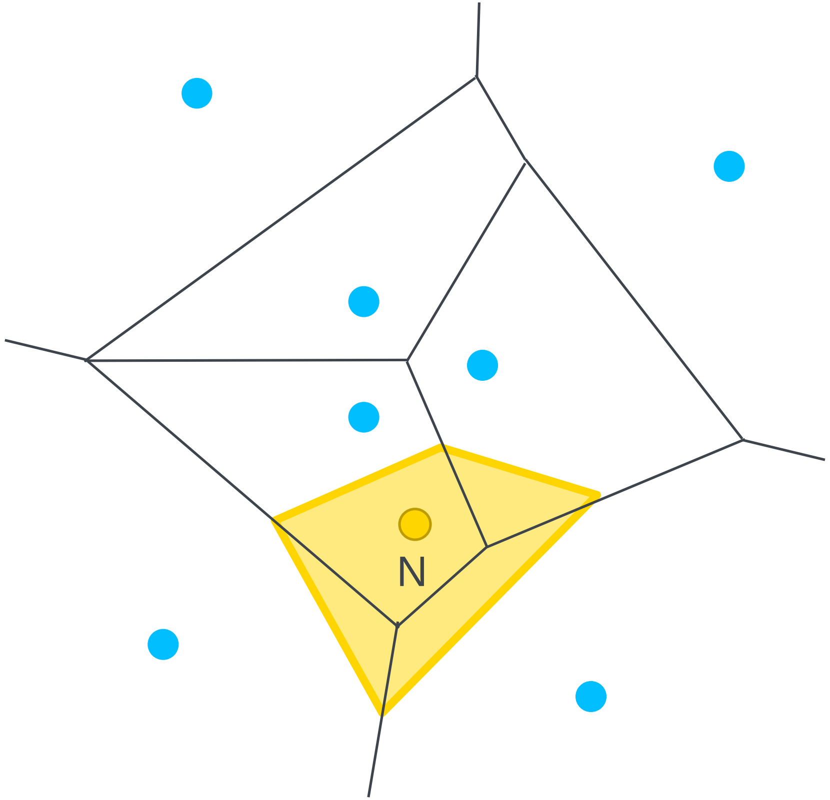
Natural-Neighbor-Based Image Reconstruction

- Natural neighbor interpolation [Sibson, 1980]
  - Smooth approximation
  - Local neighbors
  - Generally C_1 continuous
  - Voronoi



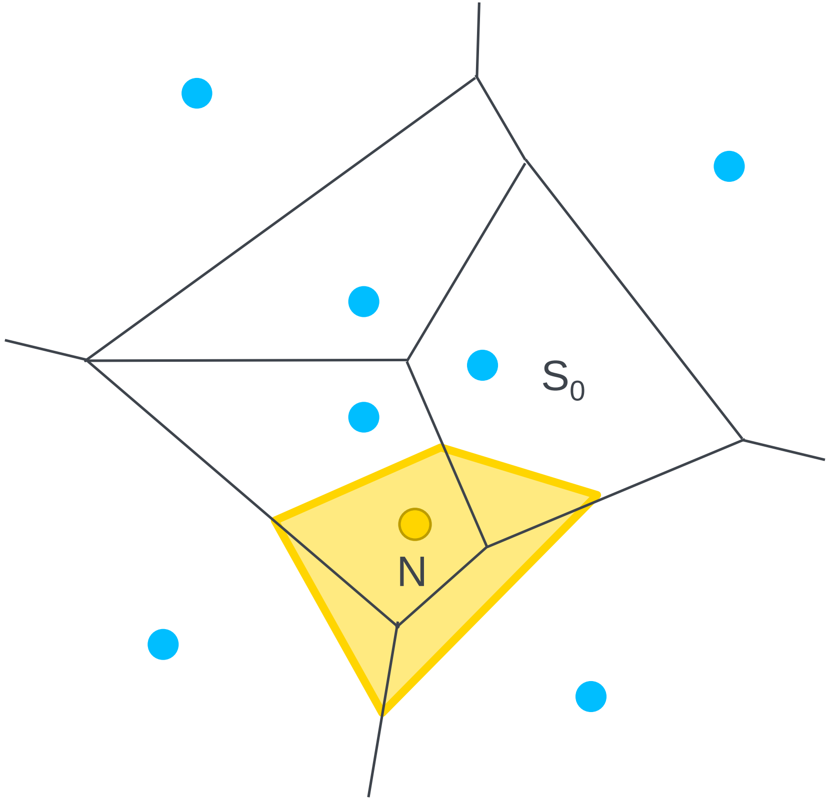
Natural-Neighbor-Based Image Reconstruction

- Natural neighbor interpolation [Sibson, 1980]
  - Smooth approximation
  - Local neighbors
  - Generally C_1 continuous
  - Voronoi



Natural-Neighbor-Based Image Reconstruction

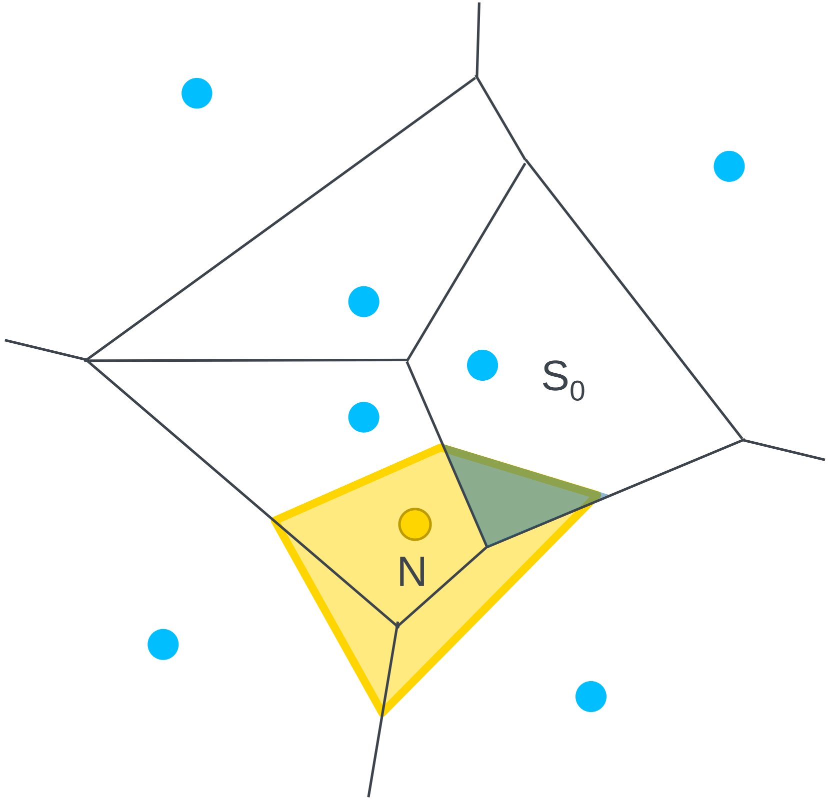
- Natural neighbor interpolation [Sibson, 1980]
  - Smooth approximation
  - Local neighbors
  - Generally C_1 continuous
  - Voronoi



$$G(x,y) = \sum_{i=0}^{k} \frac{A(S_i \cap N)}{A(N)} f(x_i, y_i)$$

Natural-Neighbor-Based Image Reconstruction

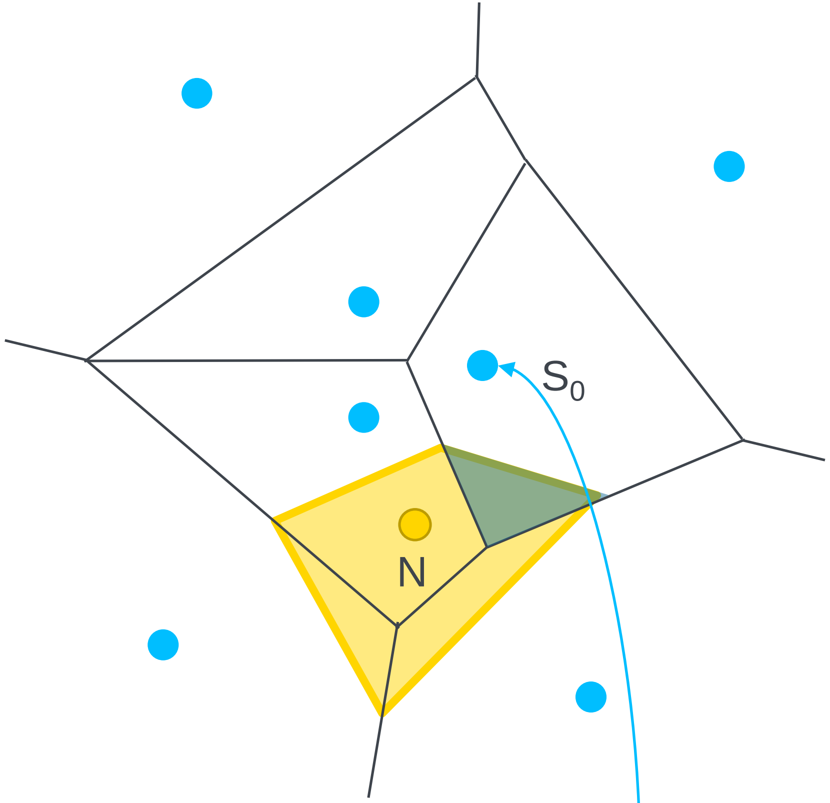
- Natural neighbor interpolation [Sibson, 1980]
  - Smooth approximation
  - Local neighbors
  - Generally C_1 continuous
  - Voronoi



$$G(x,y) = \sum_{i=0}^{k} \frac{A(S_i \cap N)}{A(N)} f(x_i, y_i)$$

Natural-Neighbor-Based Image Reconstruction

- Natural neighbor interpolation [Sibson, 1980]
  - Smooth approximation
  - Local neighbors
  - Generally C_1 continuous
  - Voronoi

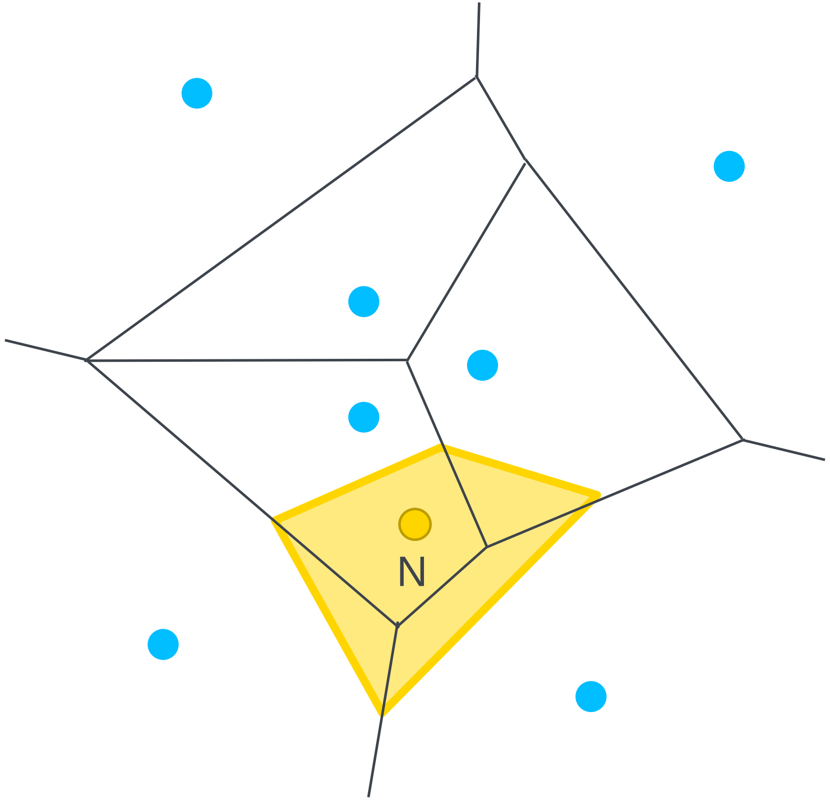


$$G(x,y) = \sum_{i=0}^{k} \frac{A(S_i \cap N)}{A(N)} f(x_i, y_i)$$

Natural-Neighbor-Based Image Reconstruction

- Natural neighbor interpolation [Sibson, 1980]
  - Smooth approximation
  - Local neighbors
  - Generally C_1 continuous
  - Voronoi

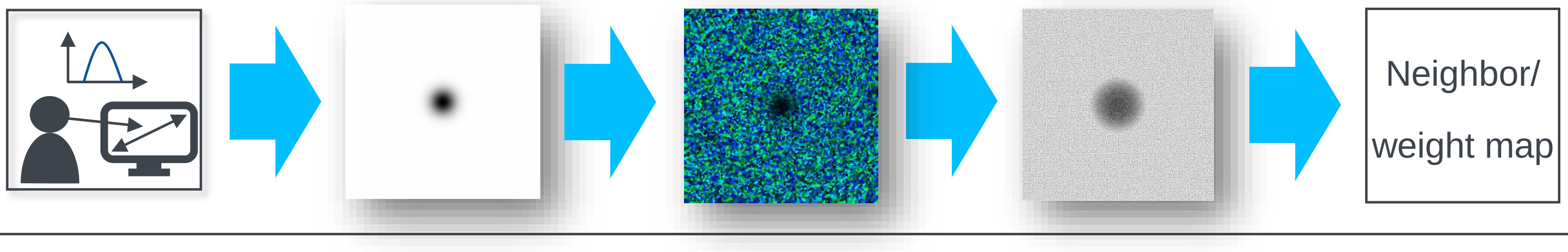
- For each pixel in sampling mask (pre-processing):
  - Save neighbor IDs and weights in map
  - Look up during runtime



$$G(x,y) = \sum_{i=0}^{k} \frac{A(S_i \cap N)}{A(N)} f(x_i, y_i)$$

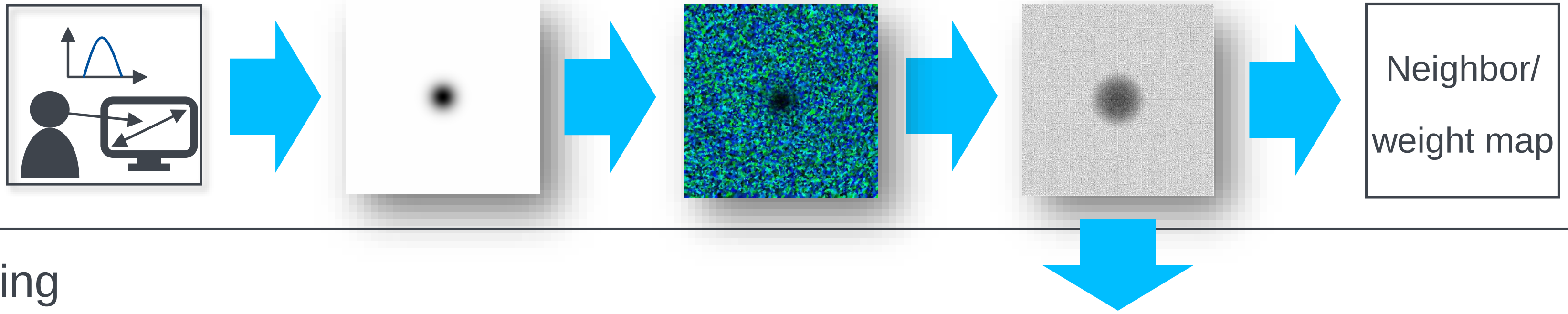
Our Approach

Pre-processing

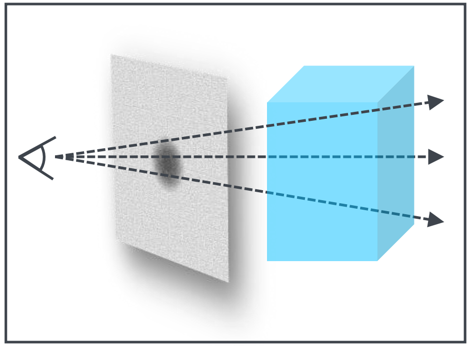


Our Approach

Pre-processing



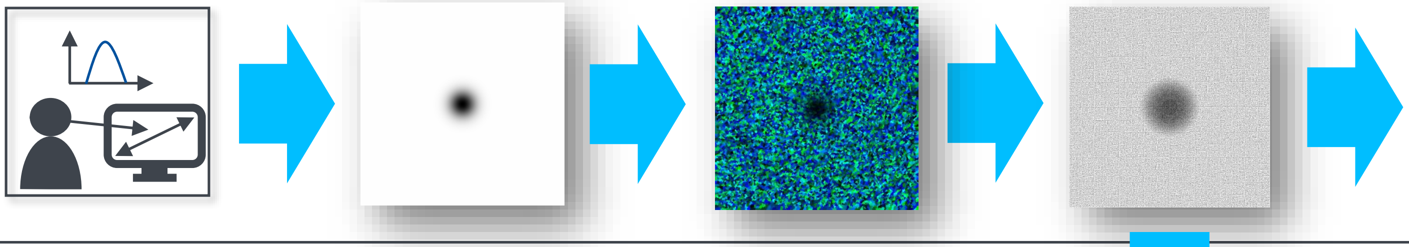
Rendering



Volume raycasting

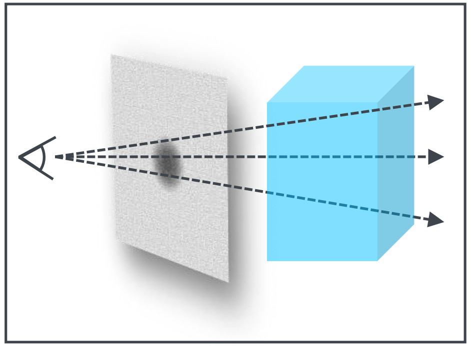
Our Approach

Pre-processing



Neighbor/
weight map

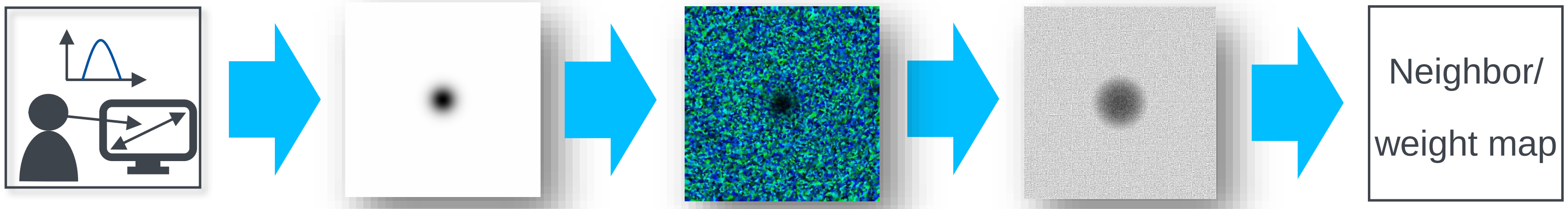
Rendering

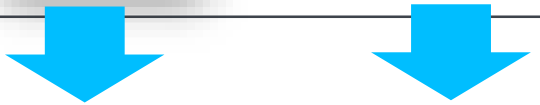


Volume raycasting

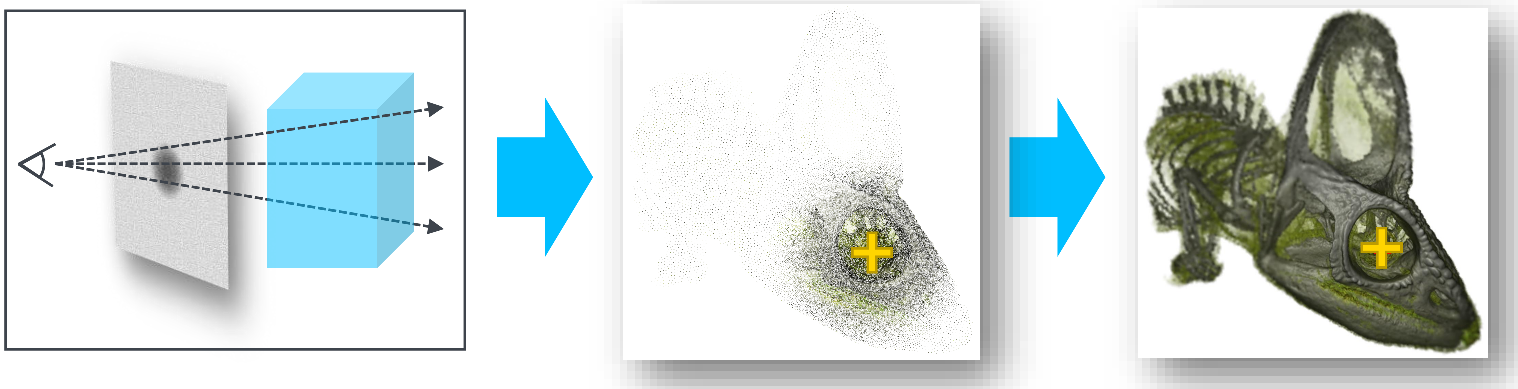
Our Approach

Pre-processing





Rendering

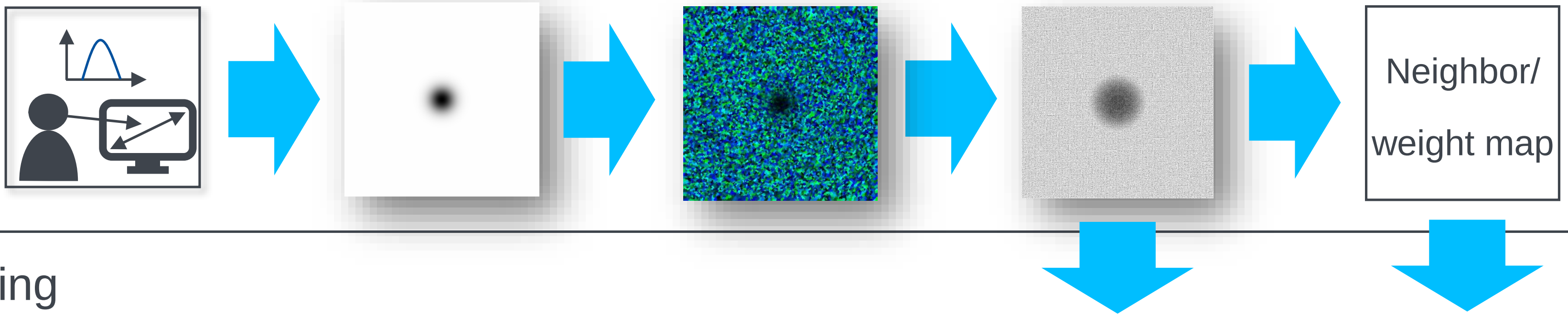


Volume raycasting

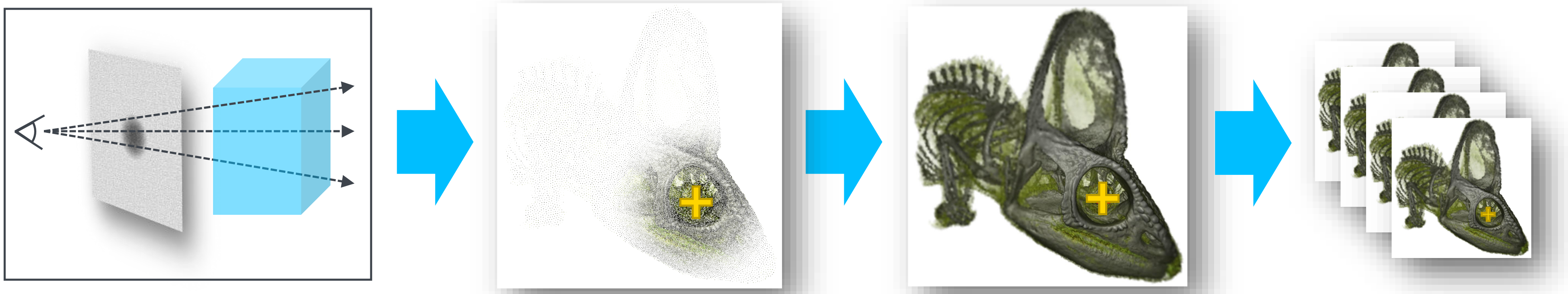
Reconstruction

Our Approach

Pre-processing



Rendering

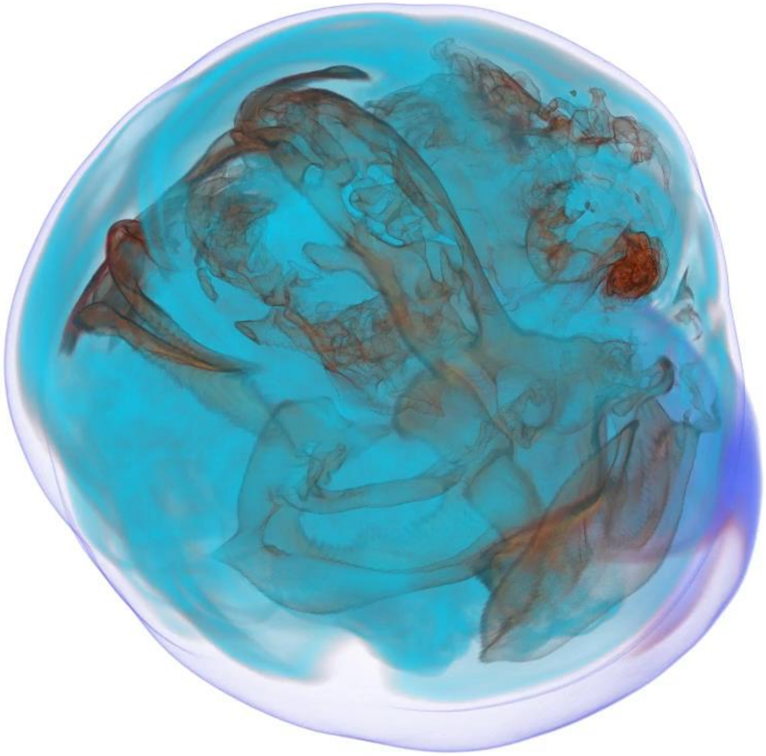


Volume raycasting

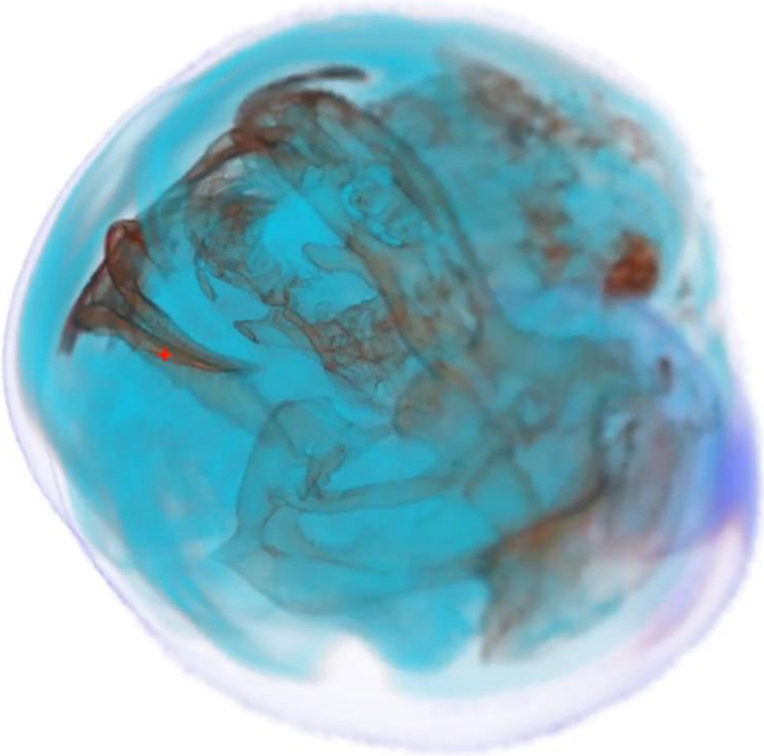
Reconstruction

Temporal smoothing

Regular: 38 fps



Foveated: 105 fps

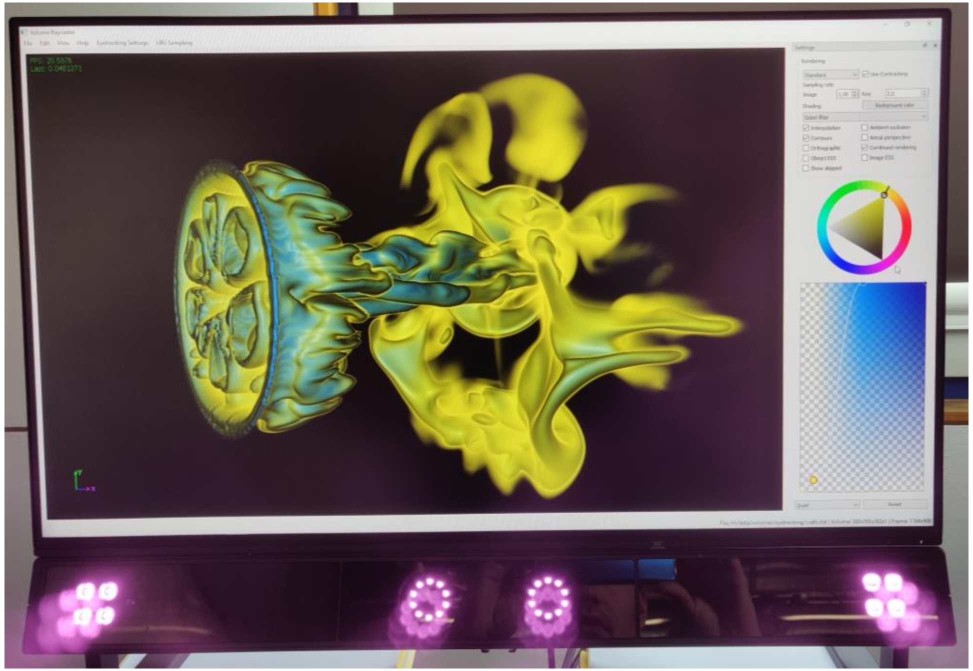


Measurements

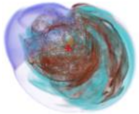
- Development: stationary Tobii Pro Spectrum eye tracker (1200 Hz)

- Automated performance evaluation
  - 1024^2 px on 24” screen
  - NVIDIA GeForce GTX 1070 (8GB VRAM)
  - 6 data sets
  - 65536 measurements
    - 256 randomly scattered gaze positions
    - 256 random camera configurations

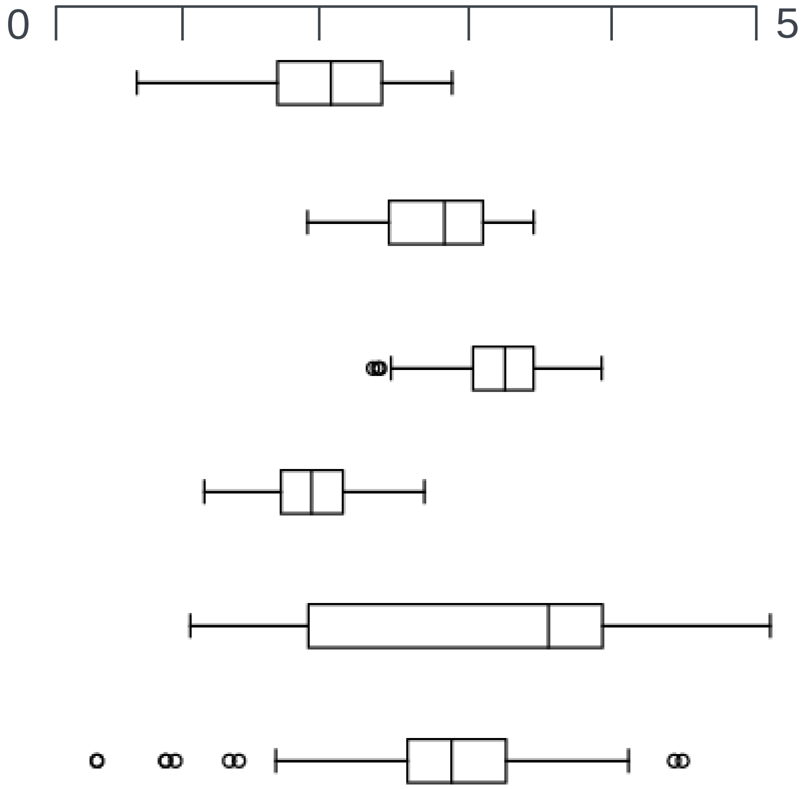
- Reconstruction overhead: 1.5 ± 0.157 ms



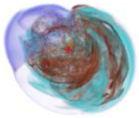
Results

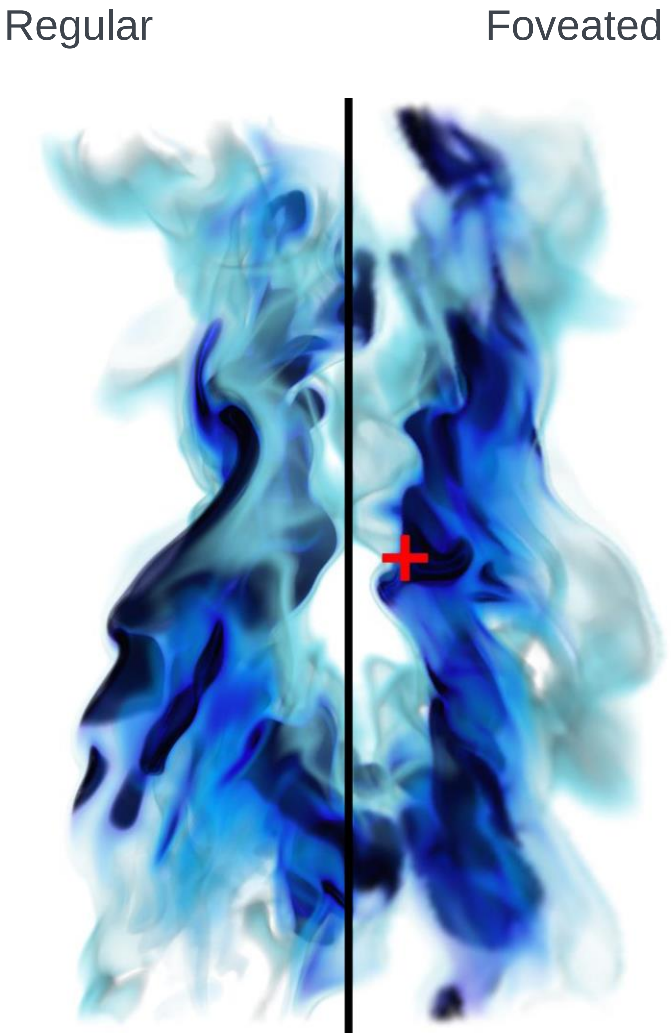
			Mean fps	
	Data set	Resolution	Regular	Foveated
	Combustion	480 x 720 x 120	73.85	154.20
	Supernova	432 x 432 x 432	37.60	105.28
	Vortex cascade	529 x 529 x 529	33.27	104.87
	Zeiss	680 x 680 x 680	98.50	177.09
	Flower	1024 x 1024 x 1024	22.79	74.08
	Chameleon	1024 x 1024 x 1080	35.99	99.78

Speedup

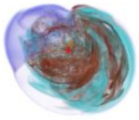


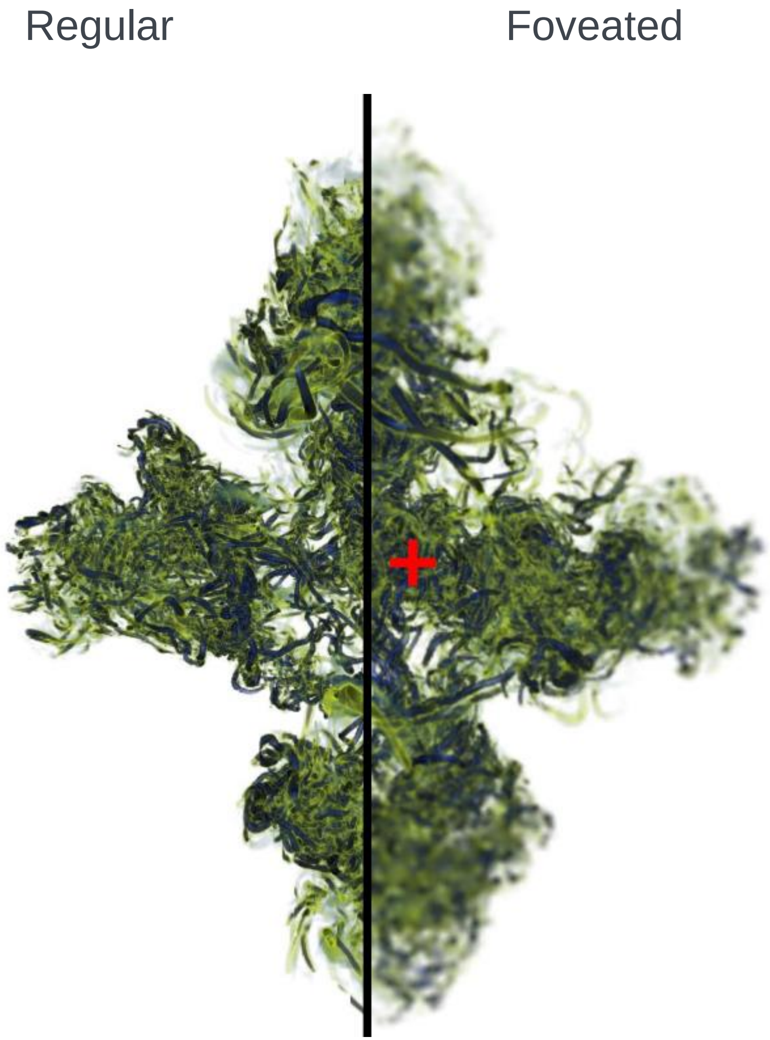
Results

			Mean fps	
	Data set	**Resolution**	**Regular**	**Foveated**
	Combustion	480 x 720 x 120	73.85	154.20
	Supernova	432 x 432 x 432	37.60	105.28
	Vortex cascade	529 x 529 x 529	33.27	104.87
	Zeiss	680 x 680 x 680	98.50	177.09
	Flower	1024 x 1024 x 1024	22.79	74.08
	Chameleon	1024 x 1024 x 1080	35.99	99.78



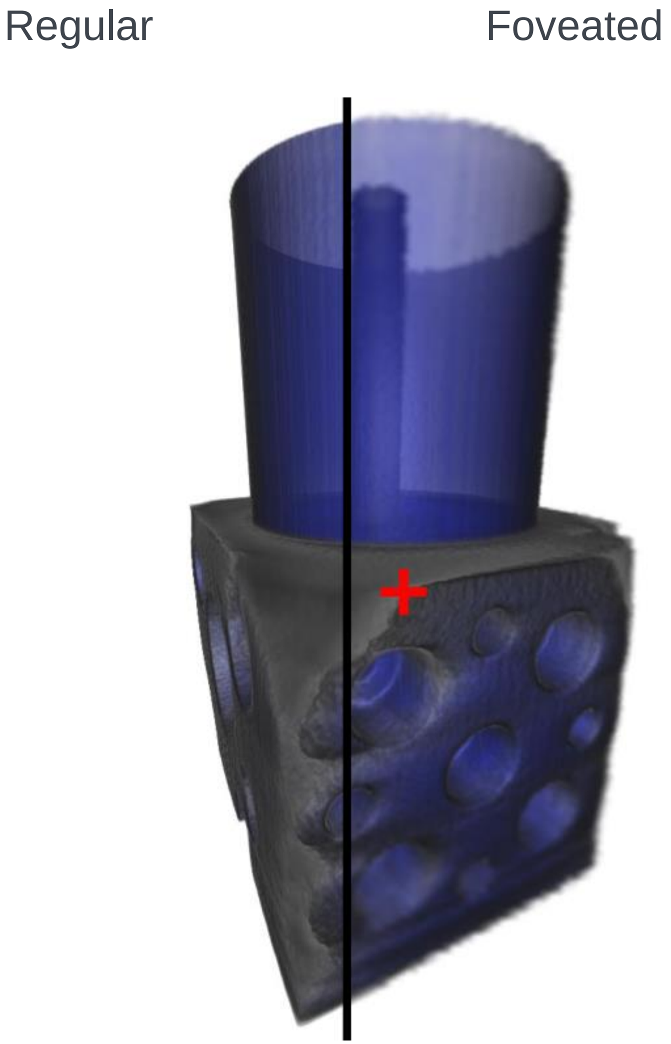
Results

			Mean fps	
	Data set	**Resolution**	**Regular**	**Foveated**
	Combustion	480 x 720 x 120	73.85	154.20
	Supernova	432 x 432 x 432	37.60	105.28
	Vortex cascade	529 x 529 x 529	33.27	104.87
	Zeiss	680 x 680 x 680	98.50	177.09
	Flower	1024 x 1024 x 1024	22.79	74.08
	Chameleon	1024 x 1024 x 1080	35.99	99.78



Results

		Mean fps	
Data set	**Resolution**	**Regular**	**Foveated**
Combustion	480 x 720 x 120	73.85	154.20
Supernova	432 x 432 x 432	37.60	105.28
Vortex cascade	529 x 529 x 529	33.27	104.87
Zeiss	680 x 680 x 680	98.50	177.09
Flower	1024 x 1024 x 1024	22.79	74.08
Chameleon	1024 x 1024 x 1080	35.99	99.78



Conclusion and Future Work

- Accelerate volume rendering by utilizing
  - Eye tracking
  - Acuity fall-off of the visual system
- Sampling: Linde-Buzo-Gray algorithm
- Reconstruction: natural neighbor interpolation
- Average speedups: 1.8 – 3.2
- Barely perceptible quality impact

Future work

- Port to head-mounted devices with integrated eye tracking
- Performance characteristics of stereo volume rendering

University of Stuttgart
Visualization Research Center (VISUS)

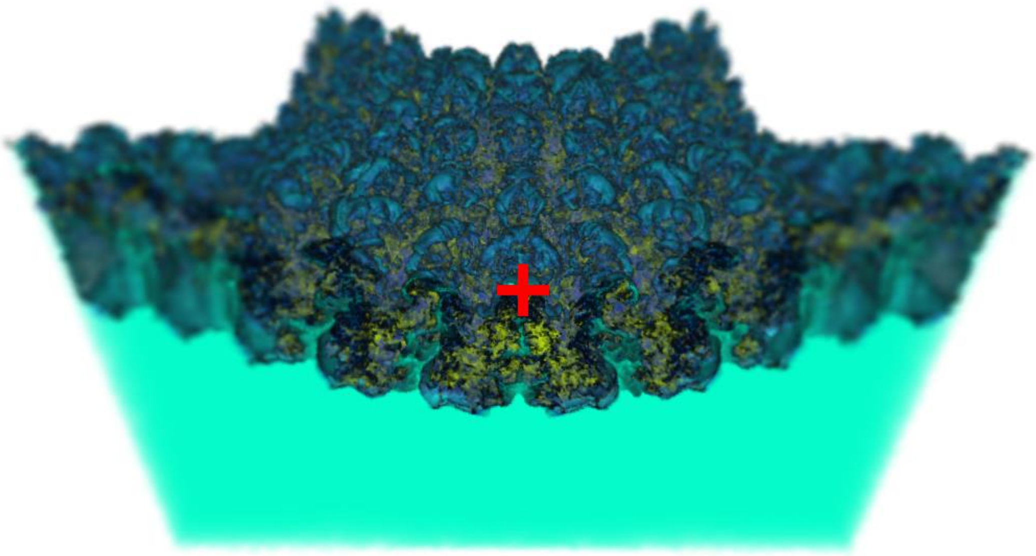
SFB-TRR 161
Quantitative Methods for Visual Computing

Thank you!

Valentin Bruder

valentin.bruder@visus.uni-stuttgart.de

https://vbruder.github.io



Visualization Research Center
University of Stuttgart

References

- BRUDER V., MÜLLER C., FREY S., ERTL T.: On evaluating runtime performance of interactive visualizations. IEEE Transactions on Visualization and Computer Graphics (2019), 1–1. 1, 3

- DEUSSEN O., SPICKER M., ZHENG Q.: Weighted Linde-BuzoGray stippling. ACM Transactions on Graphics 36, 6 (2017), 1–12. 2
- GÖRTLER J., SPICKER M., SCHULZ C., WEISKOPF D., DEUSSEN O.: Stippling of 2D scalar fields. IEEE Transactions on Visualization and Computer Graphics (2019). 2

- SIBSON R.: A vector identity for the Dirichlet tessellation. In Mathematical Proceedings of the Cambridge Philosophical Society (1980), vol. 87, Cambridge University Press, pp. 151–155. 3