VISUS (Visualization Research Center)

T. Ertl D. Weiskopf C. Dachsbacher

CUDA-Accelerated Continuous 2D Scatterplots

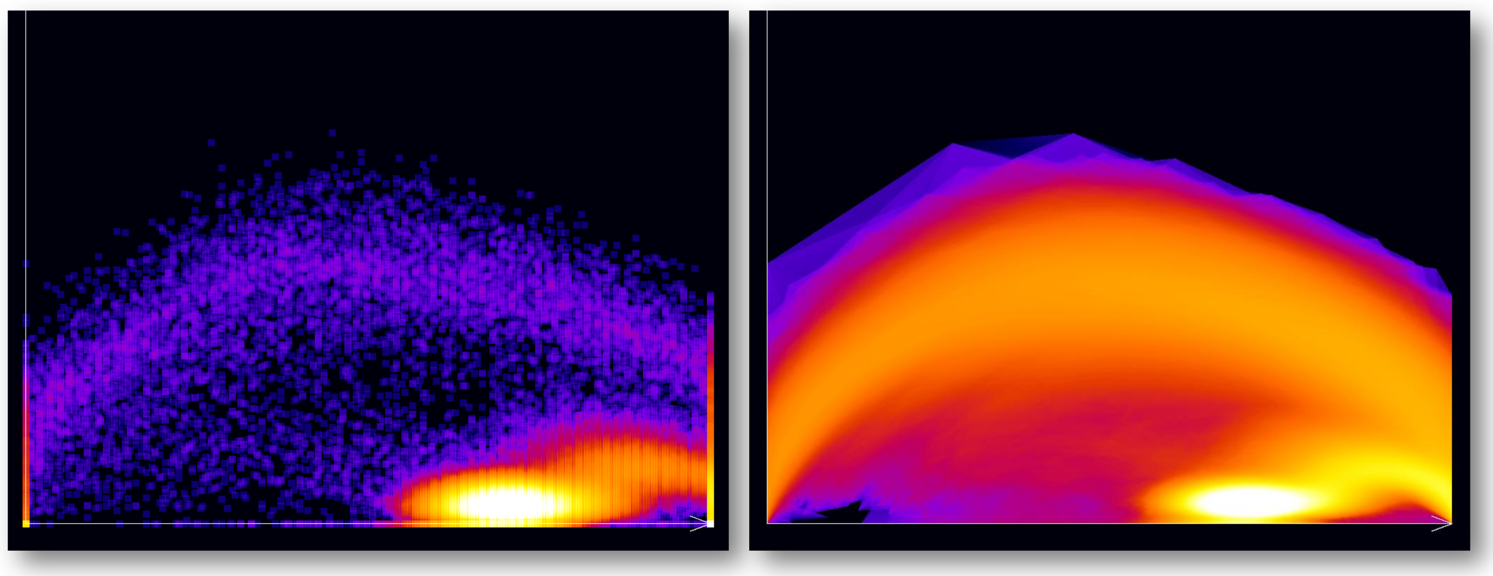
Sven Bachthaler, Steffen Frey, and Daniel Weiskopf

VisWeek 09
VIS • INFOVIS • VAST

Universität Stuttgart

Motivation

- Discretized data in scientific visualization is commonly defined continuously, e.g. by applying interpolation or reconstruction techniques to the data samples.
- Continuous scatterplots make use of continuously defined data by drawing the scatterplot in a dense way — instead of drawing discrete glyphs as it is the case with conventional scatterplots, they represent the continuous distribution function in the scatterplot domain.
- The original approach [BW08] implemented on the CPU is slow, which makes it difficult to work interactively with continuous scatterplots (e.g. increasing resolution or changing focus).



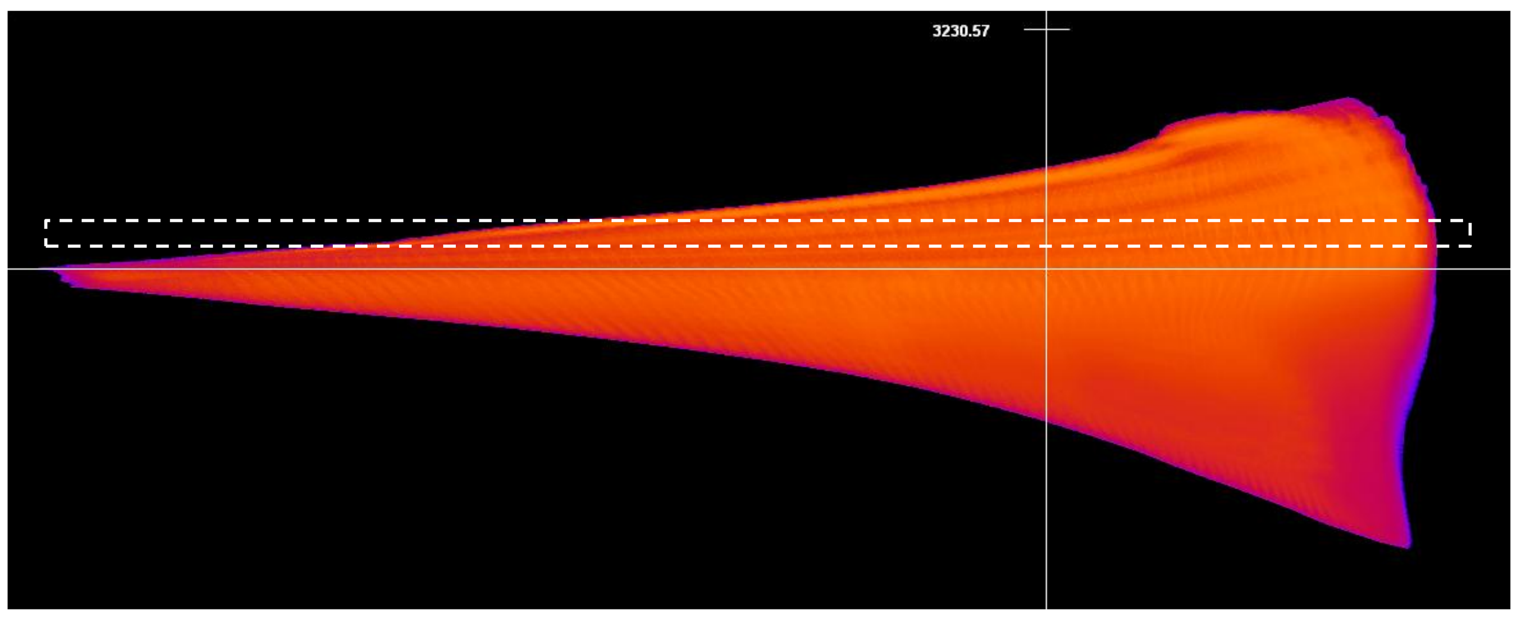
Conventional discrete scatterplot (left image) and continuous scatterplot (right image) for the tooth data set.

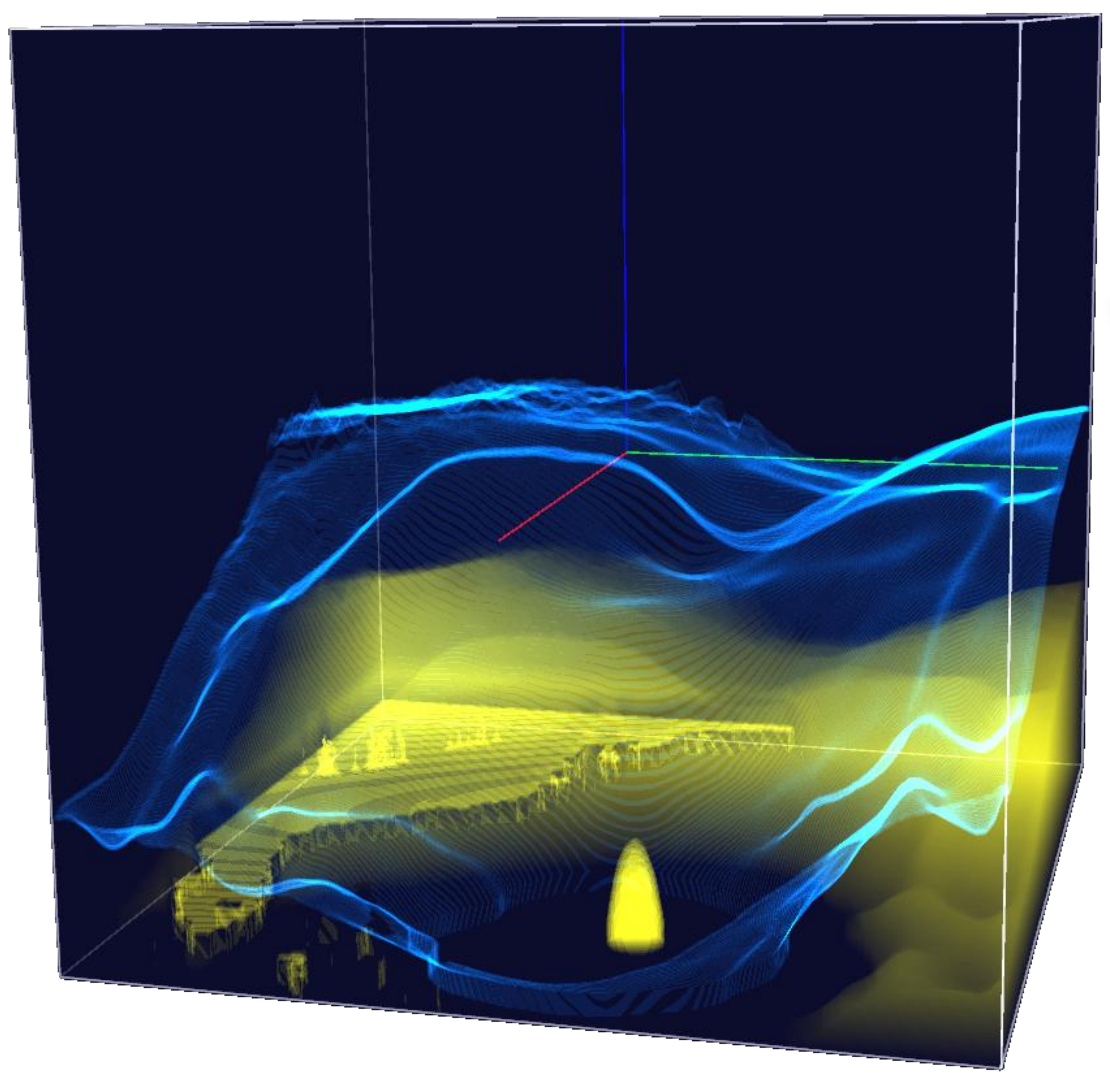
Goals

- Push performance of continuous scatterplots towards interactivity.
- Move workload from CPU to GPU.
- Improve scalability with regard to increased data set size.
- Increase usability by reducing response time.

Application

- Continuous scatterplots are identical to conventional discrete scatterplots in the limit process of infinitely dense sample points.
- Finding regions of interest in the continuous scatterplot by zooming and panning requires recomputation.
- Significantly reduced computing time motivates the user to explore the data set.
- Brushing and linking allows exploration of data sets in an efficient way.
- Example:
  - Continuous scatterplot for Hurricane Isabel data set.
  - Air temperature is mapped to horizontal axis, air pressure to vertical axis.
  - White selection box indicates brushed area, which is highlighted in the volume rendering as a transparent blue region.

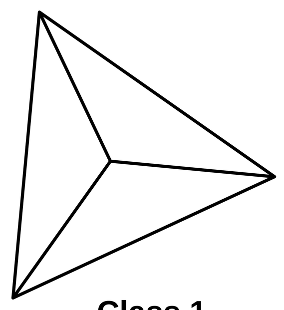




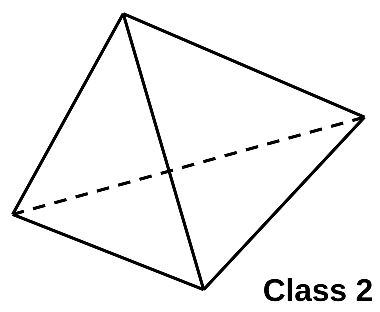
Brushing and linking

Implementation

- Pre-sorting of tetrahedra based on class of projection.
  - Class 1 and 2 (see illustration) are of relevance only, additional classes exist, but are degenerate cases of classes 1 and 2.
- Subdivide data set to fit in GPU memory.
  - Blocks process only one class of tetrahedra to ensure conherence of threads within a warp.
- Project tetrahedra based on algorithm by [WMFC02].
  - Find suitable triangle topology for projected tetrahedron.
  - Compute density for each tetrahedron.
  - Bottleneck of original CPU implementation since this step requires many computations.
- Improved kernel performance due to pre-sorted tetrahedra.
  - Diverging threads in a warp force the whole warp to serially execute each branch pass.
- Compose continuous scatterplot by drawing each projected tetrahedron using additive blending.



Class 1



Class 2

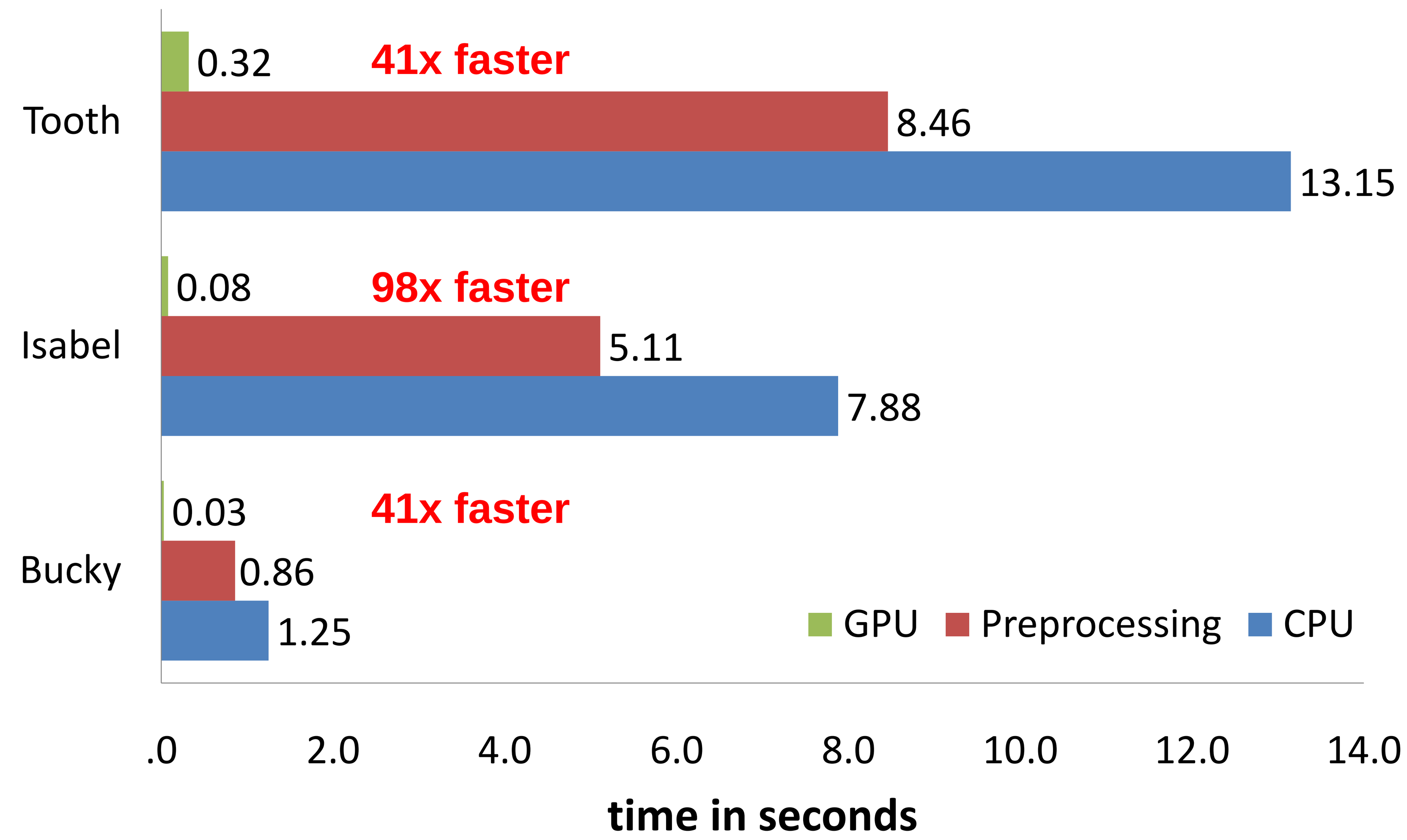
Once per data set:
- Pre-sort tetrahedra
- Upload data subsets

For each update request:
- Project tetrahedra and compute density
- Draw continuous scatterplot

CPU GPU

Performance

- Comparison of CPU and GPU implementations – time in seconds to compute a continuous scatterplot.
- Preprocessing step for GPU version only
- CPU: Intel 2.4 GHz
- GPU: Nvidia GeForce 8800 GTX
- Speed-up ranging from 40x – 100x



References

[BW08]
S. Bachthaler and D. Weiskopf:
Continuous scatterplots.
IEEE Transactions on Visualization and Computer Graphics,
Vol. 14, No. 6, pp. 1428–1435, 2008.

[WMFC02]
B. Wylie, K. Moreland, L. A. Fisk, and P. Crossno:
Tetrahedral Projection using Vertex Shaders.
In Proc. IEEE Volume Visualization and Graphics Symposium 2002, pp. 7–12, 2002.

Sven Bachthaler
bachthaler@visus.uni-stuttgart.de

Steffen Frey
steffen.frey@visus.uni-stuttgart.de

Daniel Weiskopf
weiskopf@visus.uni-stuttgart.de

Nobelstraße 15 • 70569 Stuttgart • GERMANY