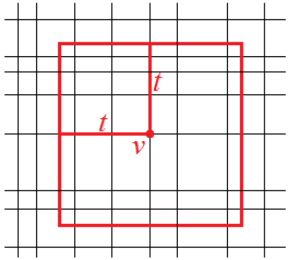
NYU TANDON SCHOOL OF ENGINEERING

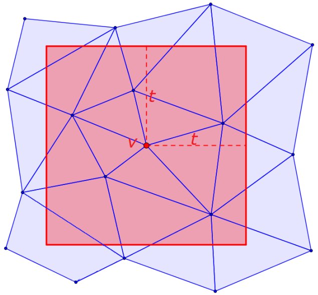
Efficient Local Statistical Analysis via Point-Wise Histograms in Tetrahedral Meshes and Curvilinear Grids

Bo Zhou, **Yi-Jen Chiang,** and Cong Wang

New York University, NY, USA

Local Histogram Computation





Given:

- Volumetric datasets, scalar or vector fields.

Obtain:

- Point-wise local histograms, computed from local regions of mesh vertices.
  - Compute at each vertex v.
  - Use local neighborhood box of fixed size t around v (same t for each vertex).

Motivation:

- Distributions are essential for analysis and visualization of large-scale data.
- Local histograms are important to study local features, and have many applications

Previous Related Work

- **Many applications for histograms**
  E.g., viewpoint selection [Takahashi *et al.* 01], identifying material interface [Thompson *et al.* 11], transfer function design [Lundström *et al.* 06], [Maciejewski *et al.* 09], [Roettger *et al.* 05], [Selver *et al.* 09.], feature tracking [Gu et al. 11], streamline placement [Xu *et al.* 10], hixels [Thompson *et al.* 11].
- **Relationship between histograms and isosurface statistics**
  (regular grids) [Carr *et al.* 06], [Scheuermann *et al.* 08], [Duffy *et al.* 13]
  (* **Continuous scatterplot** [Bachthaler *et al.* 08]: whole cells & scalar fields only)
- **Efficient computation of histograms**
- + GPU-based parallel computation [Nugteren *et al.* 11], [Scheuermann *et al.* 07]
  + Integral histograms with discrete wavelet transform [Lee *et al.* 13]
  + Computation for rectilinear grids [Chaudhuri *et al.* 12] (**)

Previous methods are mainly for regular grids or rectilinear grids (**) only
--- methods for tetrahedral meshes or curvilinear grids are lacking.

Our New Contributions

Novel **theory & algorithms** to compute point-wise local histograms for **tetrahedral meshes** & **curviliner grids** ---

- Novel sampling methods for both mesh types.
- Provably accurate method for tetetrahedral scalar fields.
- Novel overall algorithms (basically a single main algorithm)
  + theoretically sound & efficient
  + practically effective & fast
  + work for both mesh types, for both scalar & vector fields.
- Utility case study for tetetrahedral vector field visualization.

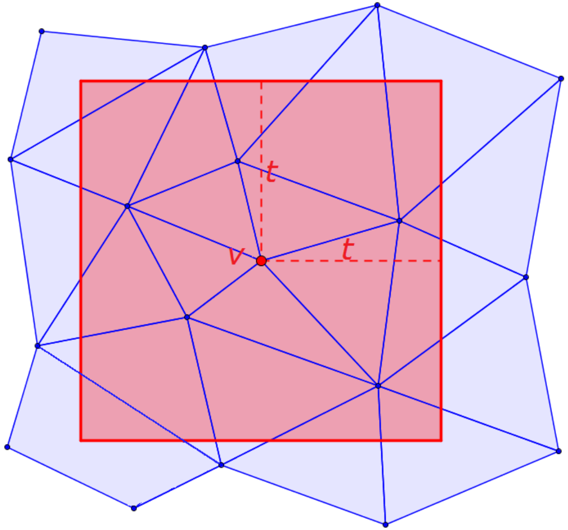
Sampling for Local Histograms

- **Box Sampling:**

Intuitively, we could generate $k \times k \times k$ samples regularly (evenly spaced) in the neighborhood box --- *box sampling*

For each sample point p:

1. Locate the cell containing p
2. Interpolate to get the data value at p
3. Add weight ($1/k^3$ of box volume) to histogram bin

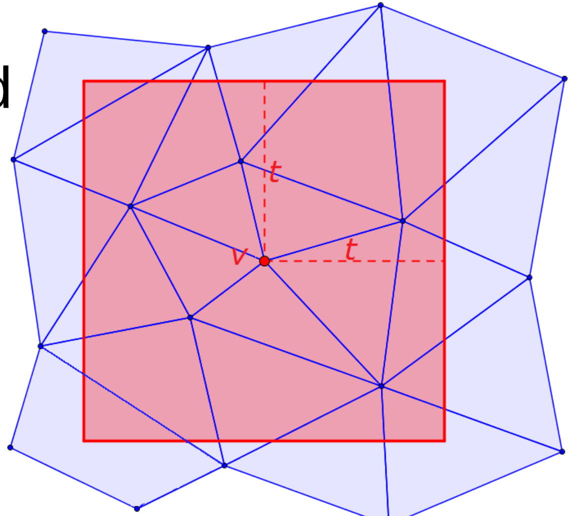


Batched cell location queries are very expensive even after decent accelerations with an octree.

Sampling for Local Histograms (Cont.)

- **Cell Sampling**:

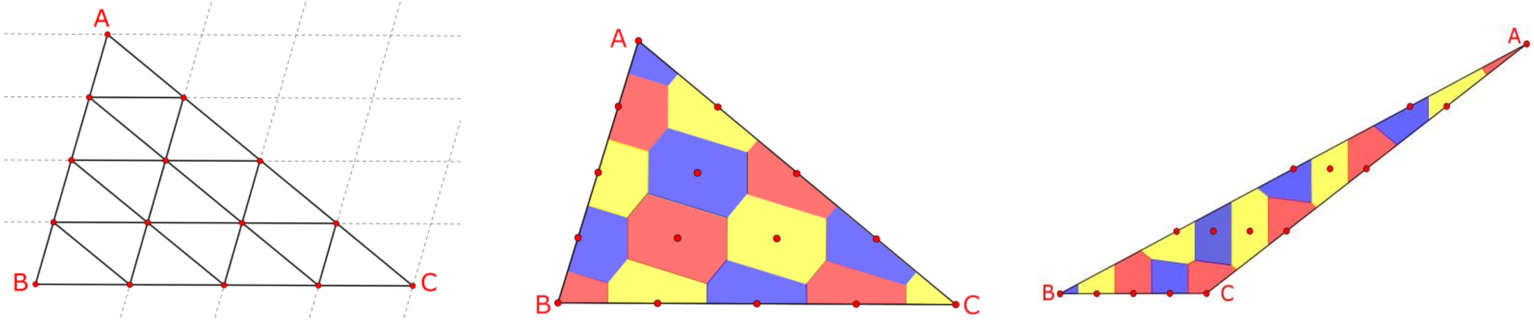
For each cell *C* intersected by the neighborhood box *N*, generate sample points in *C* and assign them to histogram bins if they lie inside *N*.
(*easy to check: N is axis-parallel*)



- Cell location queries are avoided
- Major issue: Need to assign proper weights to sample points to accurately account for their contributions.

Assigning Weights to Sample Points in Tetrahedral Meshes

- *Barycentric sampling* **–** regularly sample the cell along the barycentric axes (e.g., (B,C) and (B,A) in fig.)
- Assign weights for the sample points by their Voronoi-cell volumes?
  * Proposed in [Duffy *et al.* 13] for regular grids (easy: V / (# samples)).
  * Could be quite irregular & difficult to compute for us!



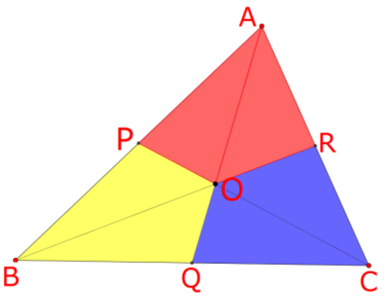
- We propose *weighting with* ***Barycentric Dual*** (def. in [Bossavit 98])

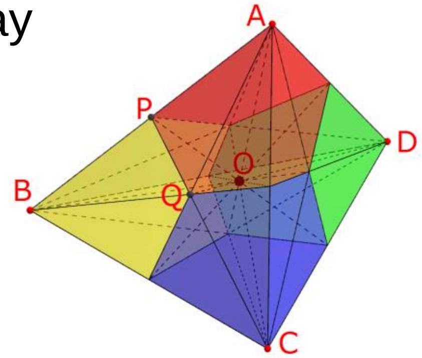
Barycentric Subdivision (BCS)

In geometry, the *BCS* is a standard way of dividing an arbitrary convex polygon/polyhedron into triangles/tetrahedra.

Divide a convex polytope into simplices of the same dimension, by connecting the barycenters of their elements of each dimension (vertex, edge midpoint, face center) in a specific way

A triangle → 6 triangles of the same area
A tetrahedron → 24 tetrahedra of the same volume





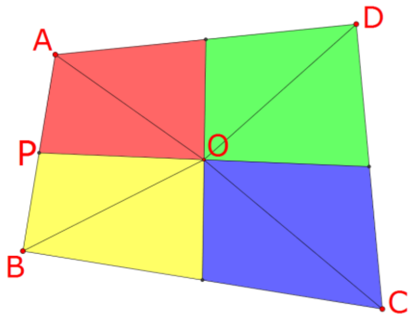
Barycentric Subdivision (BCS)

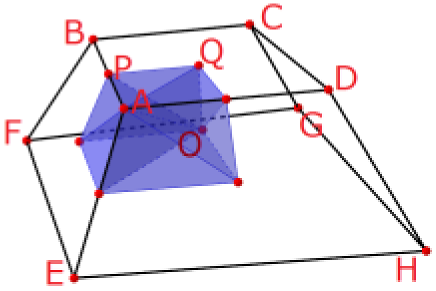
In geometry, the *BCS* is a standard way of dividing an arbitrary convex polygon/polyhedron into triangles/tetrahedra.

Divide a convex polytope into simplices of the same dimension, by connecting the barycenters of their elements of each dimension (vertex, edge midpoint, face center) in a specific way

A triangle → 6 triangles of the same area
A tetrahedron → 24 tetrahedra of the same volume

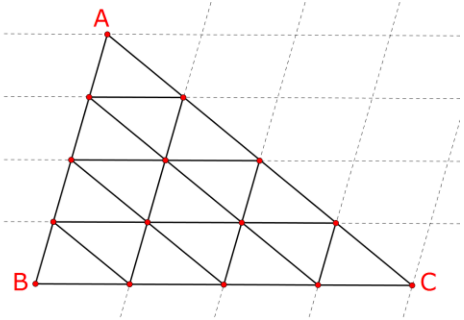
Also works for curvilinear grids (hexaheral cells)

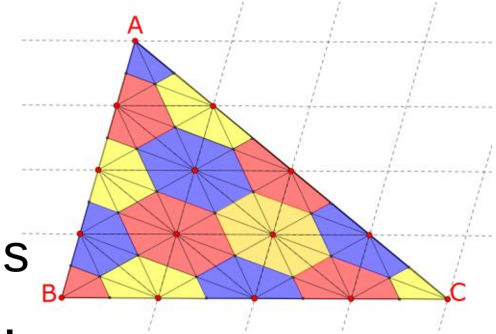




Weighting by Barycentric Dual (BD)

1. We cut cell *C* by planes that are parallel to the original faces of *C* & going thru sample points.
2. For each resulting convex polytope (triangle) we perform barycentric subdivision (BCS).
3. For each sample point *p*, we collect all final simplicies incident on *p*; the union of them is called the cell of the barycentric dual (BD) centered at *p*. (*weight of p: volume of such cell*)

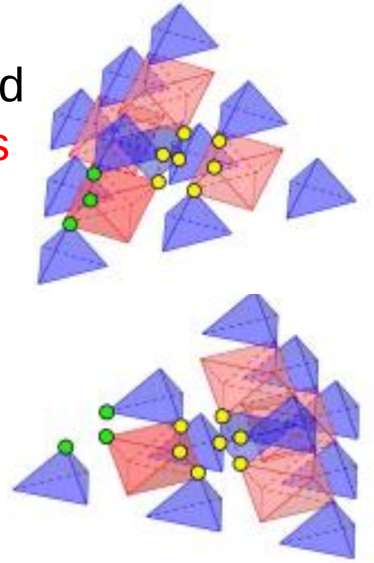




Proof of Convergence: The histogram computed this way converges to the ground truth (linear interpolant).

Geometric Properties of Barycentric Dual (BD) in Tet. Meshes

- **Geometric Properties of BD**
  The volumes of the BD cells are easy to compute. No need to actually compute BCS or BD.
  Theorem: Let V be the volume of a tetrahedral cell C, and each barycentric axis is subdivided evenly into *k* segments by k+1 samples.
  The barycentric sample points in C are of 4 types:
  (a) At the cell vertices: volume weight $1/(4k^3) \cdot V$
  (b) On the edges: volume weight $7/(6k^3) \cdot V$
  (c) On the faces: volume weight $3/k^3 \cdot V$
  (d) In the interior: volume weight $6/k^3 \cdot V$

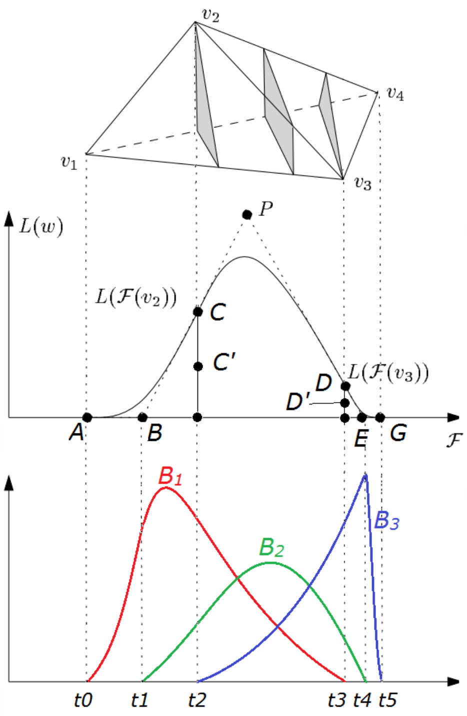


Applying Contour Spectrum

- For tetrahedral scalar fields, we can apply Contour Spectrum [Bajaj et al '97] ---

It gives a piecewise B-spline function g(h) that maps each *isovalue h* to the *accurate* area of its isosurface in a tetraheral cell C.

- Integrate g(h) on each histogram bin span w.r.t. the gradient: contribution of C to histogram bins. (Consistent with [Duffy et al. 13] using Federer's Co-Area Formula [Federer 65])
- Applicable only for each whole cell in a tetrahedral scalar field.



Applying Contour Spectrum (Cont.)

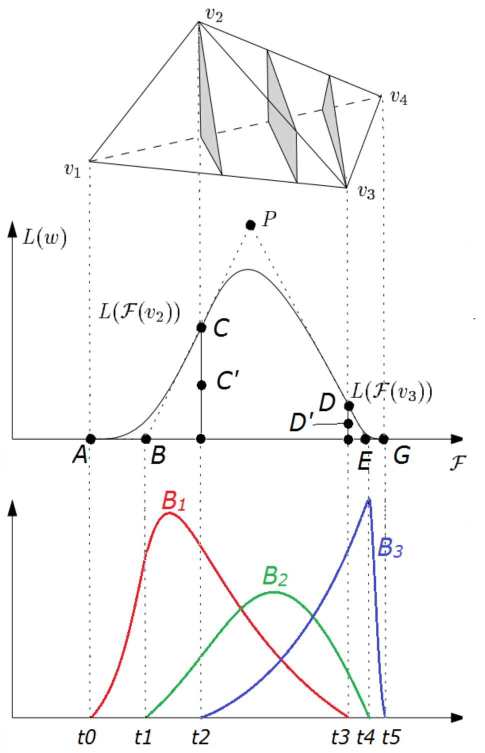
Clipping

- Contour spectrum can only apply to a *whole* cell
- For *partially intersected* cells:

  Compute and *triangulate* the intersected regions, and then apply contour spectrum on each resulting tetrahedron.

Correctness of Clipping

- Typically different triangulations can lead to different results
- We prove that they all lead to the same (and thus correct) result.
- Clipping is *provably accurate* (gives ground truth).



Efficient Algorithm for Tetrahedral Scalar Fields

- Clipping is extremely slow
  ➔ use sampling for partially intersected cells (and contour spectrum for fully contained cells)
- Sampling in a cell C may be repeated many times (once per neighborhood box partially intersecting C) ➔ Slow; avoid this!

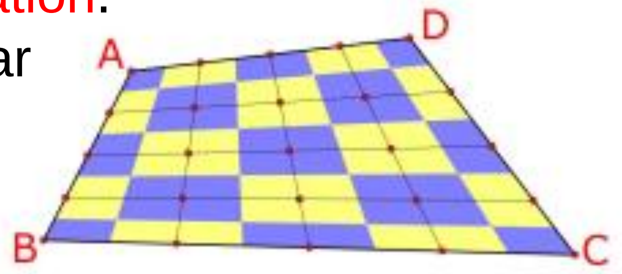
Efficient Algorithm: Cell Sampling with Sweeping

- Use a sweep plane, process each cell C (do sampling and contour spectrum on C once), in sweeping order (to be memory efficient).
- Efficiently contribute samples/contour spectrum to histograms of vertices whose boxes partially/fully contain C (using a KD-tree).
- Compute & store statistics (entropy/std. dev. etc.) at vertices for finalized local histograms to save space (Local entropy/std. dev. Field).

Local Histograms for Curvilinear Grids

Curvilinear Grids

- Use *cell sampling with sweeping* in exactly the same way.
- Contour spectrum only works for tetrahedral meshes.
  ➔ Replace it with (discrete) sampling (weighting by BD).
- The barycentric sampling only works for tetrahedral meshes.
  ➔ Do sampling similar to isoparametric interpolation.
- The volumes of the BD cells are no longer regular as in tetrahedral meshes.
  ➔ We need to compute these volumes individually (still easy, due to the BD structure).



Limitations: Each cell must be **convex** & vertices of a cell face must be **co-planar** (typically true in practice).

By-Product: Vector Fields

As in common practice (e.g., [Leopardi 07]):

- Look at vector directions.
- Use histogram bins to partition the unit sphere into angular ranges.
- Do component-wise linear interpolation on vectors (similar to vector interpolation from vertices to fragments in GPU).

Algorithm

- Use *cell sampling with sweeping* in exactly the same way.
- Contour Spectrum only works for scalar fields.
  ➔ Replace it with (discrete) sampling.

Results: Comparing Sampling Methods (I)

Comparing in Computing Global Histograms (tet. scalar fields)

- Compare: sampling with weighting by
  (1) Barycentric Dual (BD, Ours),
  (2) Voronoi Cells (VC), and
  (3) Monte Carlo sampling (MC), against
  (0) contour spectrum (ground truth).

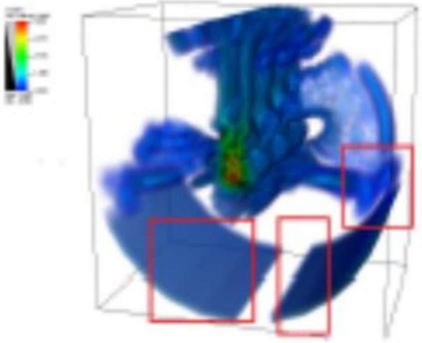
- **Summary of Results** (accuracy (NRMSE) & run-time):

1. Accuracy: VC >= Ours > MC. (MC converges very slowly.)
2. Speed: Ours > MC >> VC.
   Ours is about twice as fast as MC, and **several thousand times faster** than VC.
   * Contour spectrum is fastest (faster than generating samples).

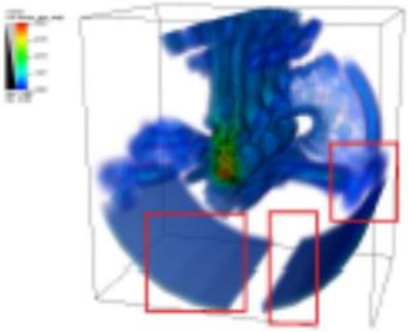
Results: Comparing Sampling Methods (II)

Comparing Under Our Overall Algorithm (tet. scalar fields, local entropy)

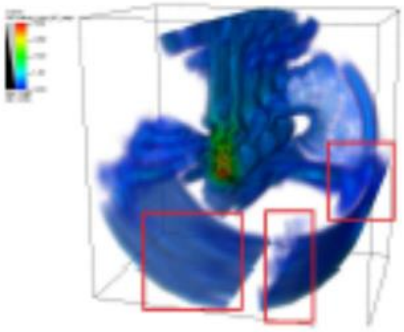
- Compare (more details in the paper):
  (1) Our sampling (Ours), and (2) Monte Carlo sampling (MC), against
  (0) Clipping (ground truth).



Clipping



Ours



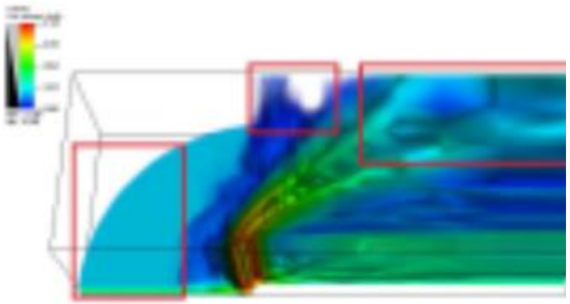
MC

- Ours can be about twice as fast as MC (e.g., 481 s vs. 934 s).

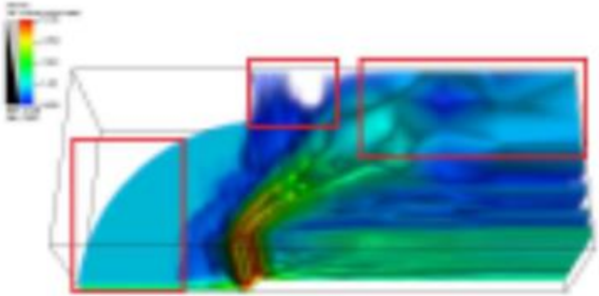
Results: Comparing Sampling Methods (II)

Comparing Under Our Overall Algorithm (tet. scalar fields, local entropy)

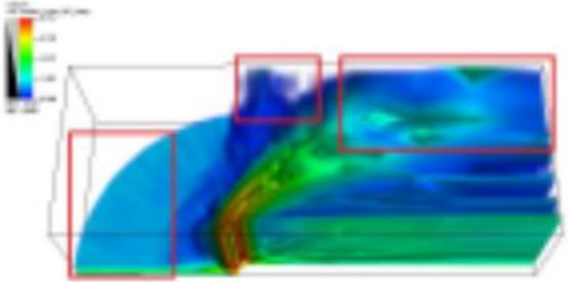
- Compare (more details in the paper):
  (1) Our sampling (urs), and (2) Monte Carlo sampling (MC), against
  (0) Clipping (ground truth).



Clipping



Ours



MC

Results: Comparing Overall Algorithms

Comparing in Computing Local Entropy/Std. Dev. (tet. scalar fields)

- Compare (more details in the paper):
  (1) Our overall algorithm cell sampling with sweeping (Ours), and
  (2) Box Sampling (with an octree for batched cell locations), against
  (0) Clipping (ground truth).

- **Summary of Results** (accuracy (NRMSE), memory & run-time):

1. Ours is about **a hundred times faster** than Clipping (e.g., **8 m vs. 11.8 h**) with very small errors.
2. Cf. Box Sampling, Ours is slightly more accurate, with much better memory usage (octree in Box Sampling can use large memory).
3. Ours is **much faster** than Box Sampling (e.g., **30 m vs. 27 h**).

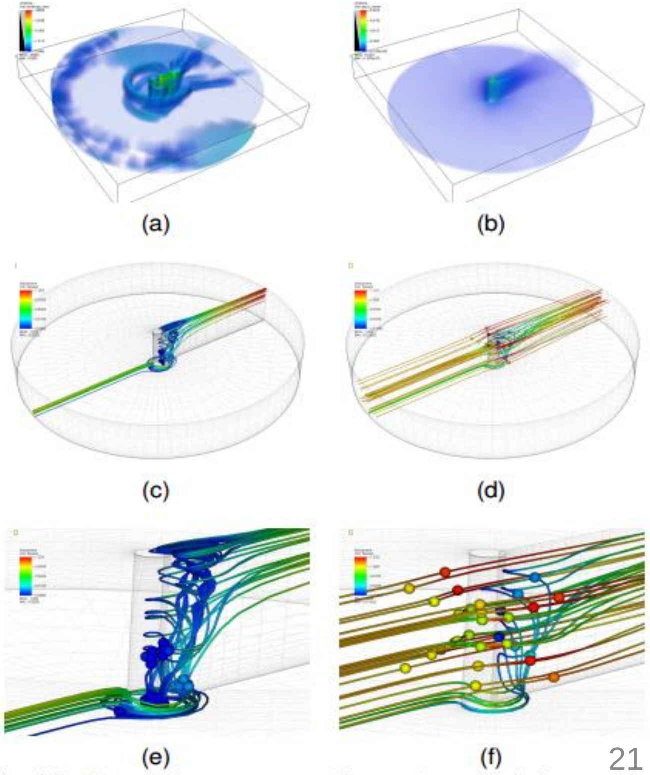
➔ **Ours should be the method of choice!**

Results: Case Study --- Tet. Vector Field Visualization

- Apply methods in [Xu *et al.* SciVis 10] --- uses local entropy field for better seeding of streamlines (regular grids only).

➔ **Now we enable them for tet. vector fields.**

Direct volume rendering on
(a) the local entropy field (Ours)
(b) the Jacobinian-norm field.
The resulting streamlines of
(c) (e) our initial seeding
(d) (f) ball seeding



Conclusions

Novel **theory & algorithms** to compute local histograms for **tetrahedral meshes** & **curviliner grids** ---

- Novel sampling methods for both mesh types: ***Barycentric Dual.***
- Provably accurate method for tetetrahedral scalar fields: ***Clipping***.
- Novel overall algorithms (**cell sampling with sweeping**)
  + work for **both** mesh types, for **both** scalar & vector fields.
  + theoretically sound & practically effective --- **method of choice** in terms of accuracy, memory, and speed.
- Utility case study for tetetrahedral vector field visualization.

Future Work and Open Questions

- Apply local entropy for transfer function design for tetrahedral and curvilinear scalar fields.
- Remove the current limitations on curvilinear grids?
- Devise a (provably) accurate method (like contour spectrum or clipping) for curvilinar grids?

Acknolwedgement:
DOE grant DE-SC0004874, program manager Lucy Nowell.

- Final version (including appendices) not in TVCG yet (DOI 1/2018) --- available at my web site (Google search on the paper title).