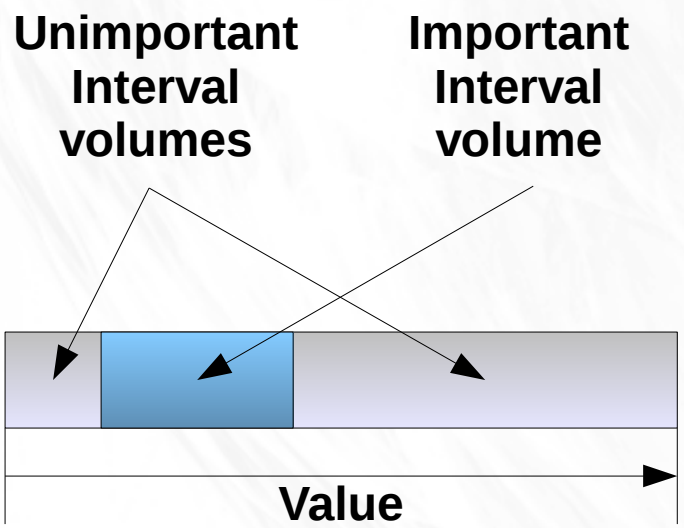
Histogram Spectra for Multivariate Time-Varying Volume LOD Selection

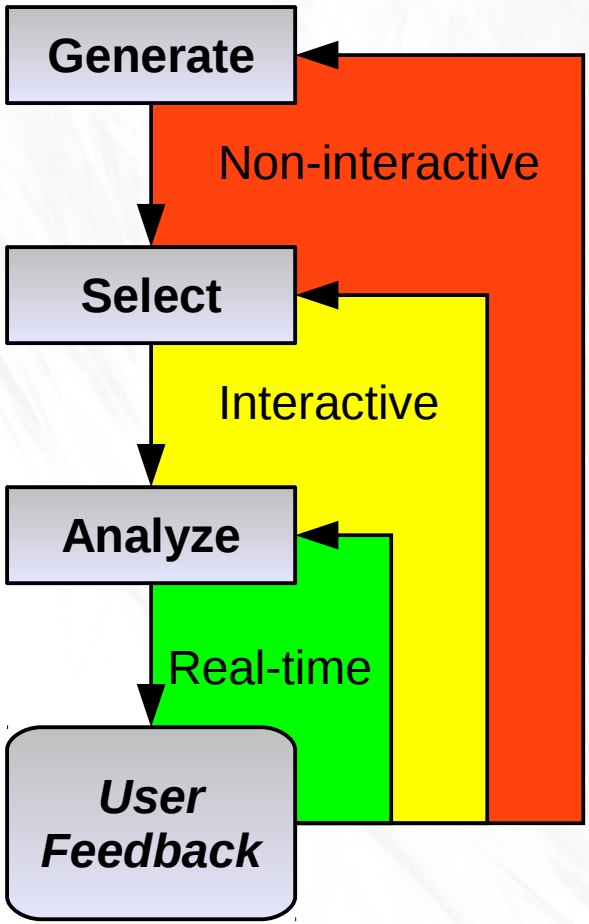
Steven Martin and Han-Wei Shen

Motivation



- Data too large to be stored in-core
- Contents of interval volumes are often of unequal importance
- Importance of interval volumes can change interactively
- Want to use memory on important interval volumes

Workflow Context



- Non-interactive data generation
  - Out-of-core storage
  - Includes simulation time
- Interactive data selection
  - Reduce data size
  - Consider needs of analysis
- Real-time analysis
  - Operates on in-core subset
  - Present results to user

LOD Selection

- Goal is to minimize error subject to a size constraint, so we need a way to estimate error
- Error estimation must be interactive because:
    - LOD selection is interactive
    - Importance of different intervals may be changed interactively.
- Data is out-of-core, so it isn't practical to directly use the data for error estimation
    - Generate metadata in noninteractive portion
    - Estimate error using the metadata

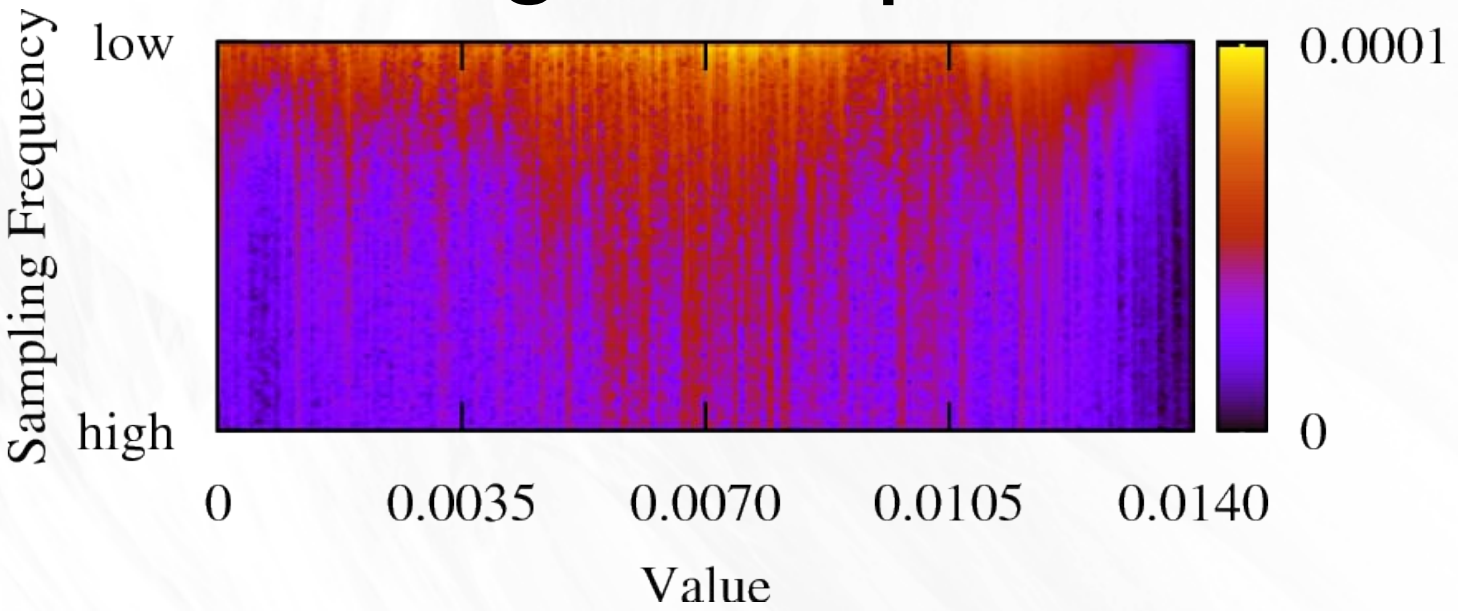
Related Work: Min-max trees

- Store the min and max values of different subregions of the volume
- Can tell us whether a part of a volume may be sensitive to LOD selection, given salient intervals, but not *how* sensitive.
- Useful for isosurfacing
  Gregorski, et al.
  “Adaptive Extraction of Time-varying Isosurfaces”
- May be possible to extend for LOD selection, but this is not the approach we take.

Contributions

- A kind of metadata, *Histogram Spectra*, is introduced to enable error estimation for interactive LOD selection on time-varying multivariate data
    - Different interval volumes may have different importance weights
    - Importance weights can be changed interactively, with the LOD optimization also being performed interactively, enabling interactive data selection
- Enables improved quality by applying optimization to approximately minimize error over salient intervals, subject to a size constraint

A Histogram Spectrum



Intuition: Stores a discretization of the differences between the PDF of the ground truth and the PDF of downsampled levels of detail.

Stored per-block

$$h(x,a) = \left| f_b(x) - f_a(x) \right| = \text{Histogram spectrum}$$

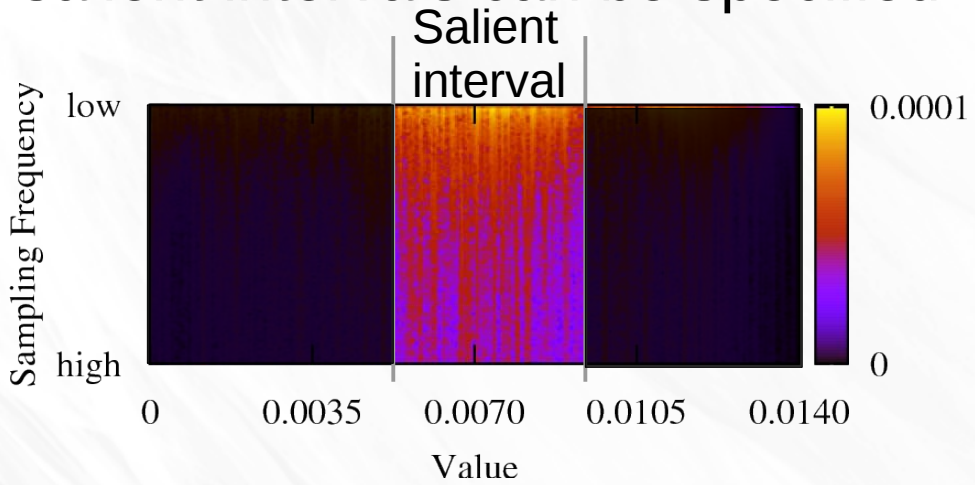
$$f_a(x) = \text{PDF of block sampled with frequency } a$$

$$f_b(x) = \text{PDF of ground truth block}$$

$$x = \text{A sample value}$$

Weighted Histogram Spectrum

- Values are not necessarily of equal importance
- To account for this, the histogram spectrum is weighted according to saliency
- Multiple salient intervals can be specified



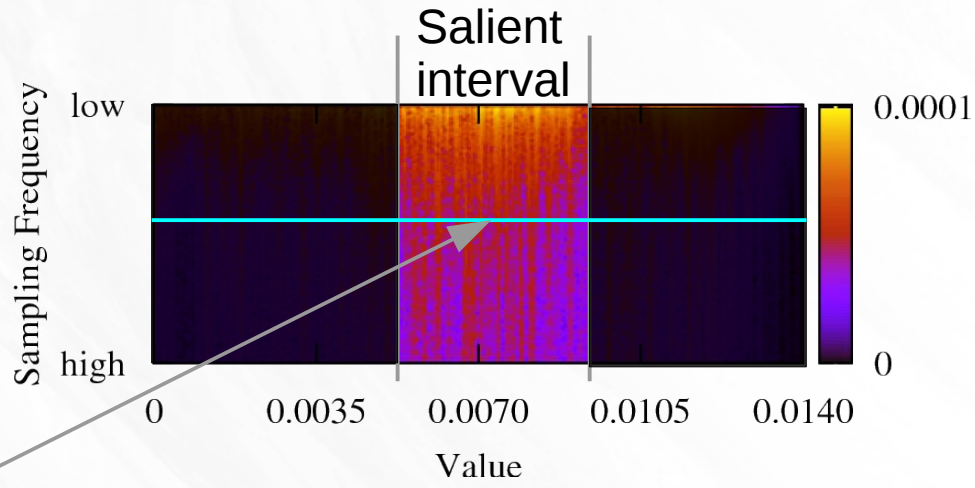
$$h_w(x,a) = w(x)h(x,a) = \text{Weighted histogram spectrum}$$

$$w(x) = \begin{cases} 1 & , x \in Y \\ 0 & , \text{otherwise} \end{cases}$$

$$Y = \text{Set of all points in the salient interval volume}$$

Error Estimation with a Weighted Histogram Spectrum

- An integration along a horizontal line segment within the weighted histogram spectrum
- Vertical position (a) determined by the LOD



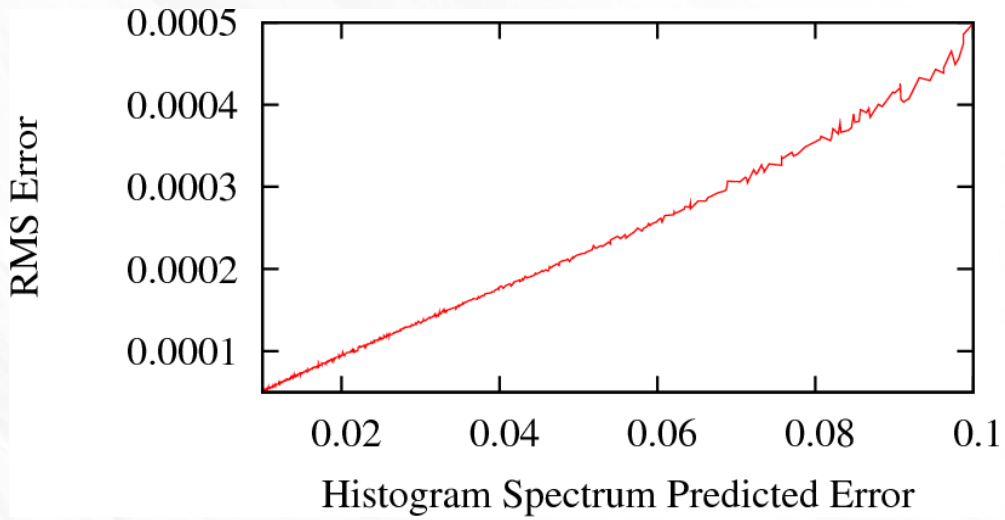
$$E(a) = \int_{-\infty}^{\infty} h_w(x, a)\, dx = \text{Histogram Spectrum Predicted Error}$$

Intuitively, this is the approximate change in the volume of the salient interval volume

Interpretation of Predicted Error

- Predicted error enables error estimation using compact precomputed metadata instead of the original data
- RMS error, *without the use of metadata*, is used for comparison; not practical for interactive workflows
- Proportionality is reasonable because downsampling tends to increase RMSE and change the volume of interval volumes

A typical result:



Applying Histogram Spectra

- Every block has a histogram spectrum
  - Metadata, much smaller than input, per-block
  - Precomputed during the non-interactive data generation phase; the dataset must be read once
  - Precomputation time is dominated by the single streaming read pass, rather than compute
- Because the histogram spectrum predicted error is proportional to the RMS error, we can use it in place of the RMS error for LOD selection.
  - Any inaccuracy will manifest itself in the error present in the results for a given size constraint

Optimization for LOD Selection

$$\underset{L}{\operatorname{argmin}} \sum_{i=1}^{N} E_i\left(a_{L_i}\right)$$

where

$$\sum_{i=1}^{N} S_{i,L_i} \le S_{\max} \quad \text{(Size constraint)}$$

$$1 \le L_i \le M \quad \text{(Level constraint)}$$

$$L_i \in \mathbb{Z} \quad \text{(Integer levels)}$$

L_i is the LOD of block i

$S_{i,k}$ is the size of block i at LOD k in bytes

$S_{\max}$ is the maximum total size

N is the number of blocks

M is the number of levels

E_i is the histogram spectrum predicted error for block i

a_{L_i} is the sampling frequency for the LOD assigned to block i

- Unknowns are level of detail assignments
- Nonlinear because E_i varies nonlinearly with a change in level

Optimization for LOD Selection

- Not practical to solve directly
  - A linear binary integer programming formulation is in the paper – still not practical
  - However, we did apply it using GLPK for evaluating accuracy.
- A greedy solution was found to be effective in our test cases.

Greedy LOD Selection

- Heuristic: Error density

$$A_{i,j} = \frac{E(a_j)}{S_{i,j}} = \text{Error density}$$

where

$E_i(a_j)$ is the predicted error in block i at LOD j

$S_{i,j}$ is the size in bytes of block i at LOD j

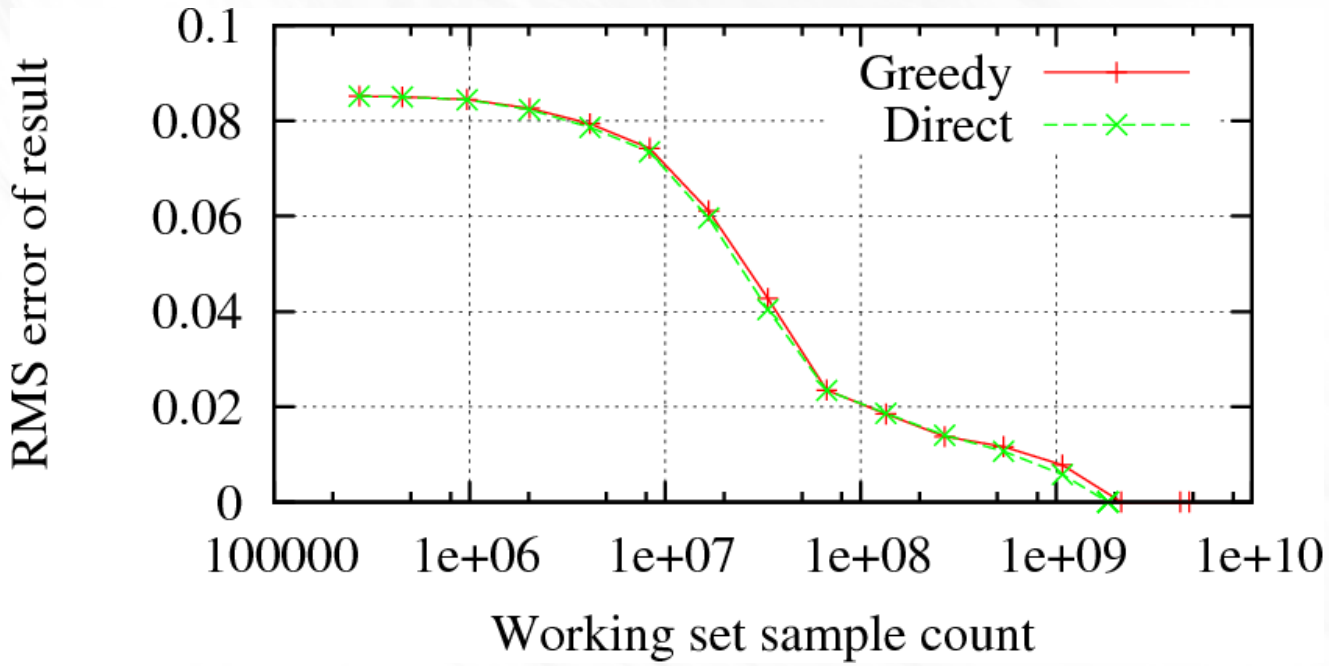
- We want blocks that have low error density (to minimize overall error for a given size)
- Apply an *evaluate – sort – select* algorithm

Evaluate – Sort – Select

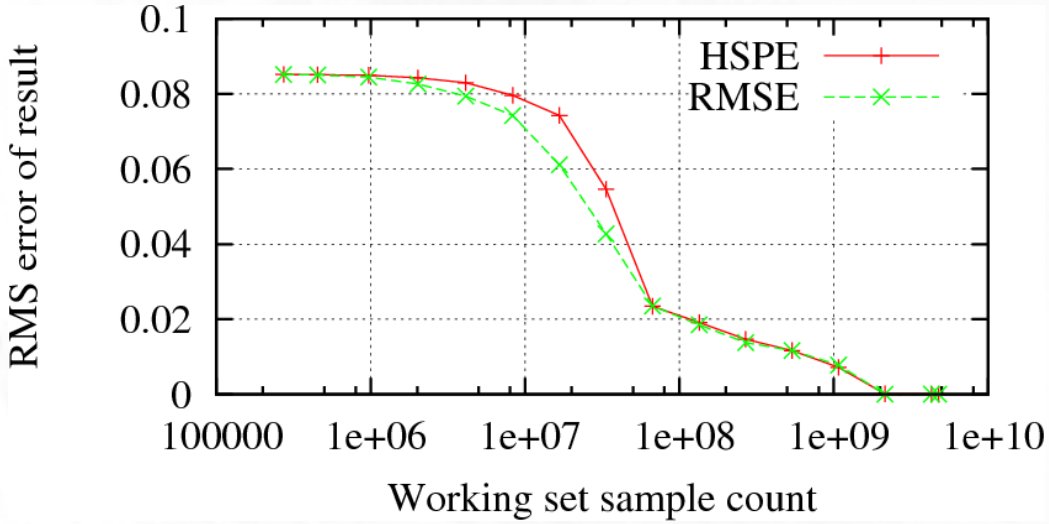
1) Create a list of potential LOD assignments
    - One entry for every block for every LOD
    - Evaluate the error density heuristic for each
2) Sort assignments in ascending order of the heuristic
3) Apply LOD assignments incrementally from the start of the list until the size constraint is reached
    - Initial assignments for blocks are at lowest LOD
    - Every time an assignment is made to a block, the total size is updated by subtracting the old assignment size and adding the new assignment size

Greedy Approximation vs. Exact

- GLPK applied using the linear binary integer programming form of the problem for a direct solution
- GLPK 1000x-100000x slower (to be expected...)
- Results nearly identical for the tested cases



RMS Error vs. Predicted Error



- Using histogram spectrum predicted error instead of RMS error within the optimization algorithm yields similar results.
- HSPE metadata usage reduces reads needed, greatly improving speed.
- This is reasonable, given the proportional relationship observed on all of our datasets
  - Volume data
  - Image data

Multivariate Extension

	Univariate	Multivariate, C variables
Objective Function	$\underset{L}{\operatorname{argmin}} \sum_{i=1}^{N} E_i(a_{L_i})$	$\underset{L}{\operatorname{argmin}} \sum_{k=1}^{C} \sum_{i=1}^{N_k} E_{k,i}(a_{k,L_{k,i}})$
Predicted Error Function	$E_i(a) = \int_{-\infty}^{\infty} h_{w_i}(x,a)\,dx$	$E_{i,k}(a) = \int_{-\infty}^{\infty} h_{w_{i,k}}(x,a)\,dx$
Weighted Histogram Spectra	$h_{w_i}(x,a) = w(x)h_i(x,a)$	$h_{w_{i,k}}(x,a) = w_k(x)h_{i,k}(x,a)$

- Handled via a summation of univariate problems
- Independent histogram spectra
- Each variable has its own weighting function

Multivariate Weighting

- Consider the case where interval $[a_0:a_1]$ of variable A is only important when it occurs within the interval $[b_0:b_1]$ of variable B
- The weighting function for A, w_A, depends on a and b, in this case.
    - The important intervals are not known a priori
    - Need interactive re-weighting; cannot go back to data
    - This would require storage of joint histogram spectra between A and B
- An alternative: *conditional importance*

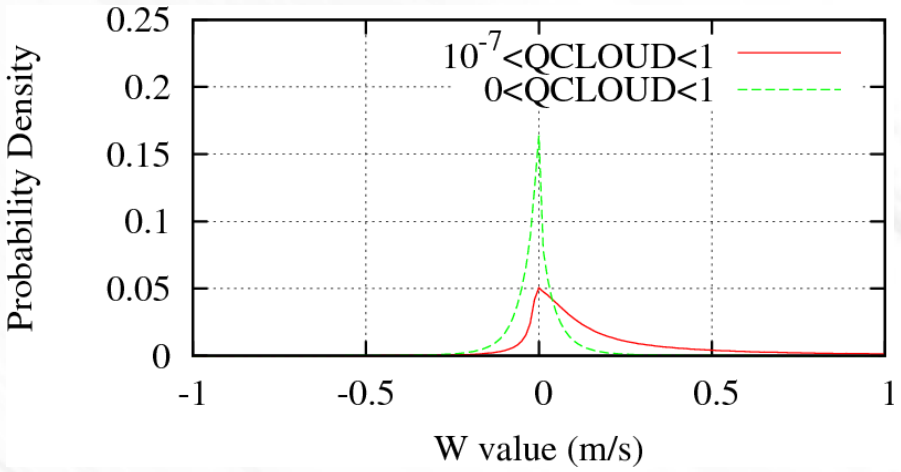
Conditional Importance

- Consider again the case where interval $[a_0:a_1]$ of variable A is only important when it occurs within the interval $[b_0:b_1]$ of variable B
- Approximate the relationship between A and B using conditional probability
- Does not require storage of joint histogram spectra
- Future work: Consider truly joint distributions

$$w_A(a) = \begin{cases} k_A P(A = a | B \in [b_0 : b_1]) & , a \in [a_0 : a_1] \\ 0 & , \text{otherwise} \end{cases}$$

k_A is a weighting constant for variable A

Example: Probability density of velocities is different within different QCLOUD interval volumes

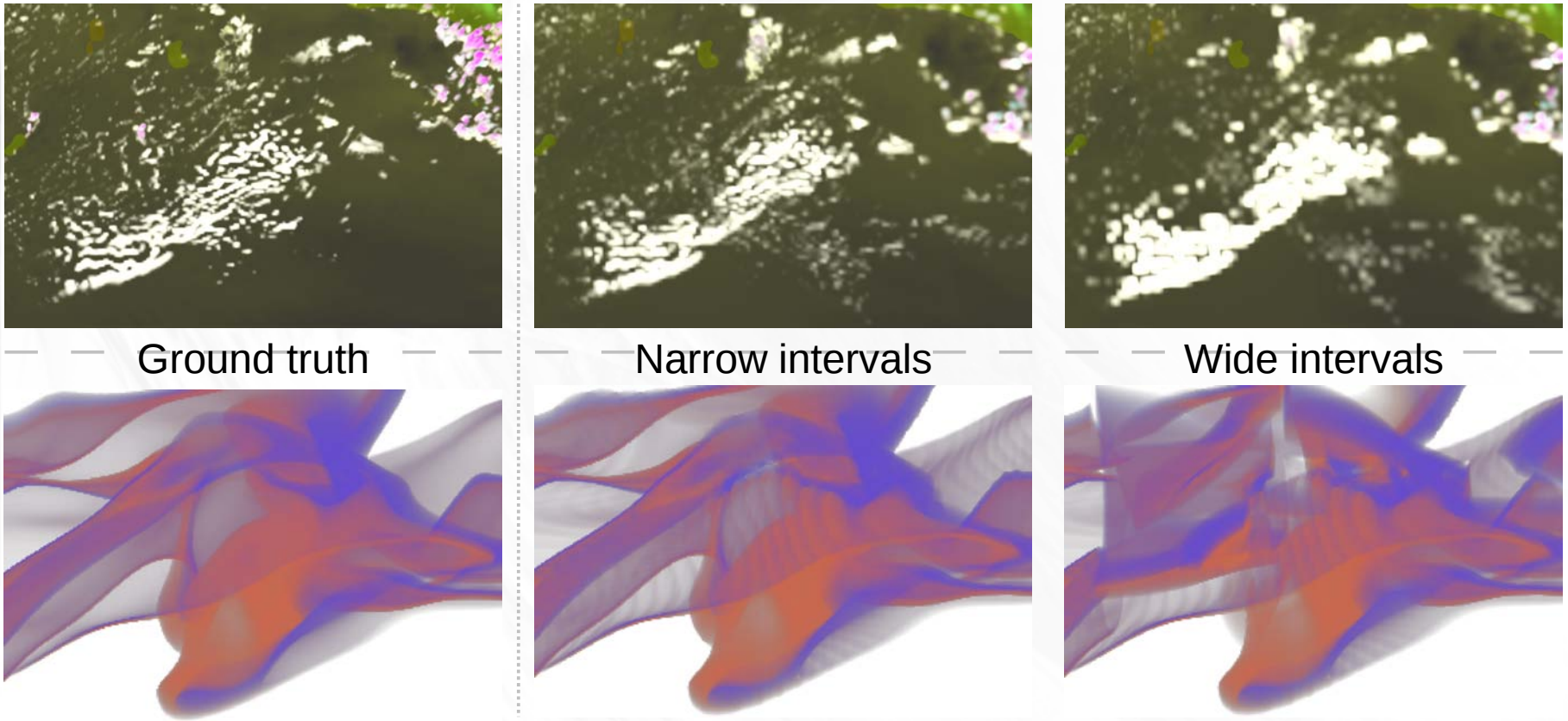


Results: Speed

	Climate	Combustion
Original data	117GiB; 8 LODs; 5,920 blocks	69GiB; 8 LODs; 17,010 blocks
Histogram spectra (metadata)	67MiB	60MiB
LOD selection time using RMS error (no metadata)	1782 seconds	3900 seconds
LOD selection time using histogram spectra predicted error (with metadata)	0.1 second	0.2 second

- Both tests used the greedy approximation algorithm
- Data size is much larger than system memory (4GiB)
- Using RMS error, disk accesses are needed, contrary to using histogram spectra predicted error

Results: Quality



- Narrower salient interval volumes tend to yield better quality: less volume to sample with same data size
- More data yields higher quality: more data to sample the same volume

Future Work

- HSPE vs. other error metrics
- Joint histogram spectra
  - Storage challenges
  - Curse of dimensionality
  - May not be much better than conditional importance
- Consider continuity constraints between blocks in time and space

Acknowledgments

This work was supported in part by the DOE Office of Science, Advanced Scientific Computing Research, under award number DOE-SC0005036, Battelle contract No. 137365, and SciDAC grant DE-FC02-06ER25779, program manager Lucy Nowell.