Reduced Representations for In-Situ Visualization

Steffen Frey and Thomas Ertl - Visualization Research Center, University of Stuttgart

Workshop on Software Frameworks for Scalable Scientific Simulations - ISC 2015
July 16th, 2015

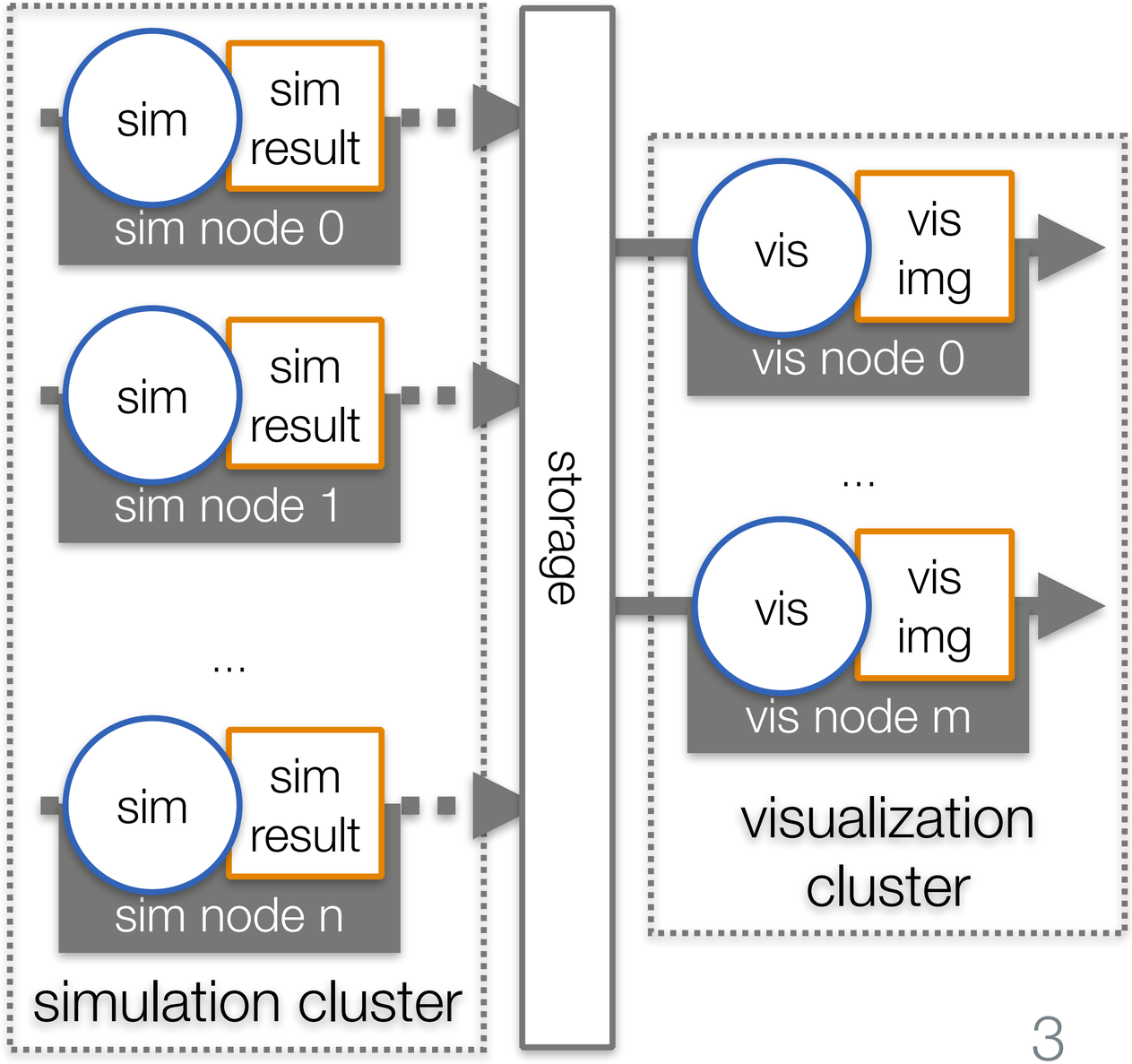
Visualization Research Center
University of Stuttgart

Motivation

- increasing dataset sizes
  - large-scale supercomputer simulations
- storing (or moving) all generated raw data impossible/ impractical
  - consumes a lot of space, time and power
  - just discarding data not desirable
- approach: in-situ (on site) reduction of data
- by transformation to visualization representation that
  - … achieves a significant reduction in size
  - … can be generated quickly/efficiently
  - … enables the analysis the selected data of interest
  - … allows for interactive exploration
  - … can be used for monitoring and steering

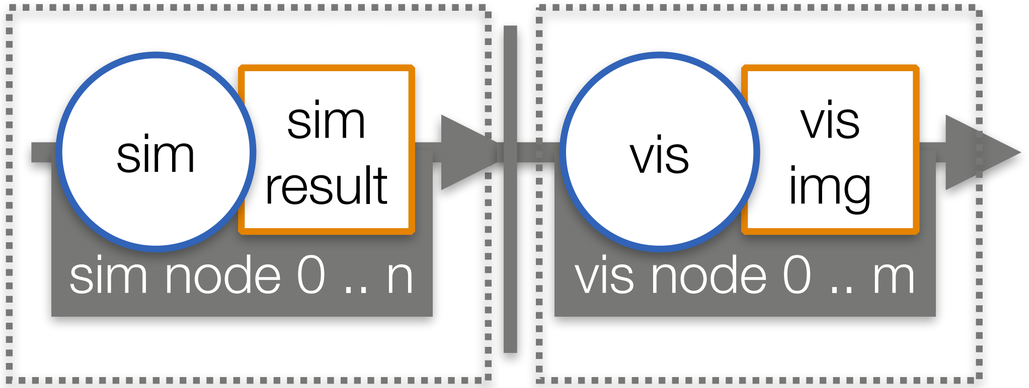
Simulation-Visualization Coupling (1)

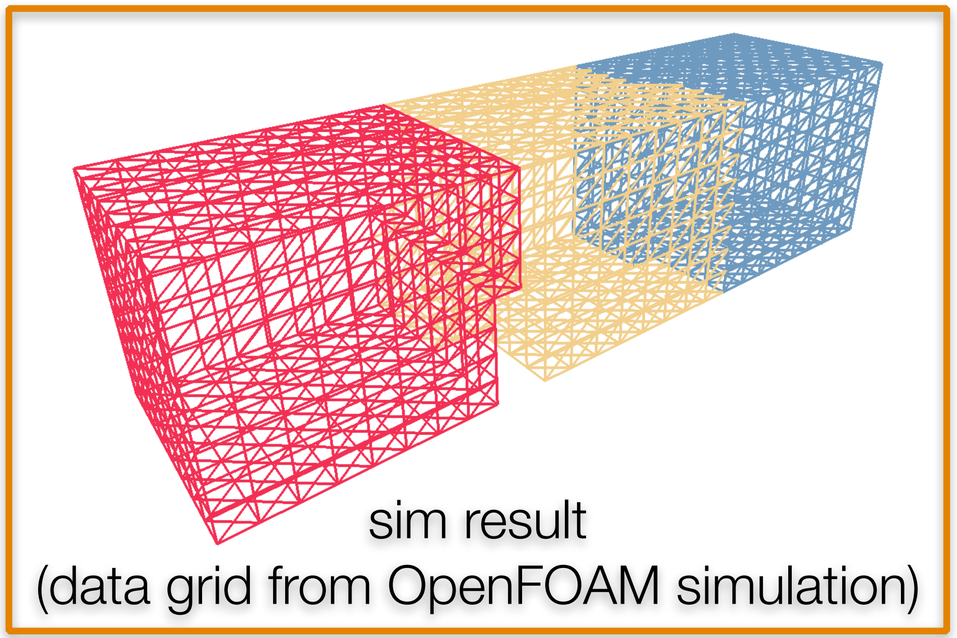
- *traditional simulation-visualization procedure*
  - decoupled **sim**ulation and **vis**ualization
    - vis runs independently after sim is completed
    - vis only needs to by able to read output data structure
  - sim result data stored directly → huge amounts of data need to be transferred
    - storage/time/energy constraints limit resolution of stored result data
      - typically w.r.t. time
  - … but full flexibility preserved for exploring/analyzing stored data



Simulation-Visualization Coupling (2)

- *loose coupling*
  - simulation and visualization run concurrently
    - sim result not stored to disk anymore
    - amount of data that is stored permanently is reduced
      - vis result should be much smaller than sim result
    - results become available during simulation run
  - … but on separate resources
    - high network load
    - forced data conversion
    - potentially redundant generation of data structures

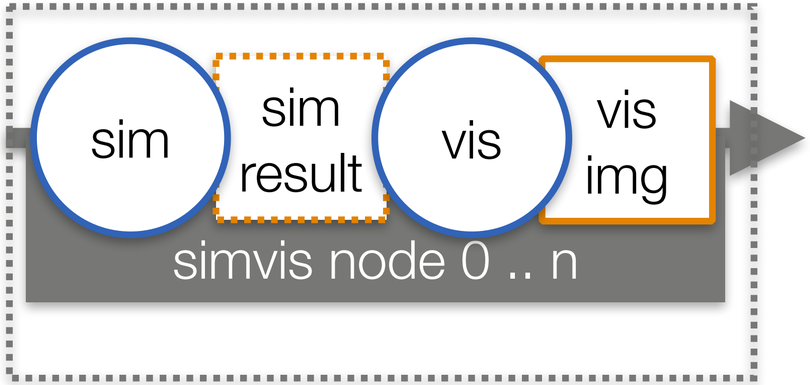


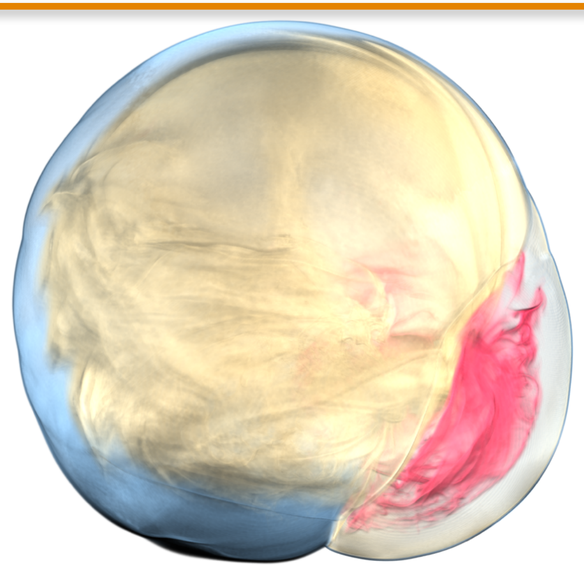


loose, tight and hybrid coupling classification according to [Riva et al.] Marzia Rivia, Luigi Caloria, Giuseppa Muscianisia, Vladimir Slavnicb “In-situ Visualization: State-of-the-art and Some Use Cases”, 2014.

Simulation-Visualization Coupling (3)

- *tight coupling*
  - visualization runs on the same cluster node as simulation (in-situ)
  - they share …
    - data structures
      - optimally no transfer or data conversion required
    - space decomposition
    - compute resources
    - …
  - the result are images generated by the visualization
    - no flexibility a posteriori for (interactive) exploration



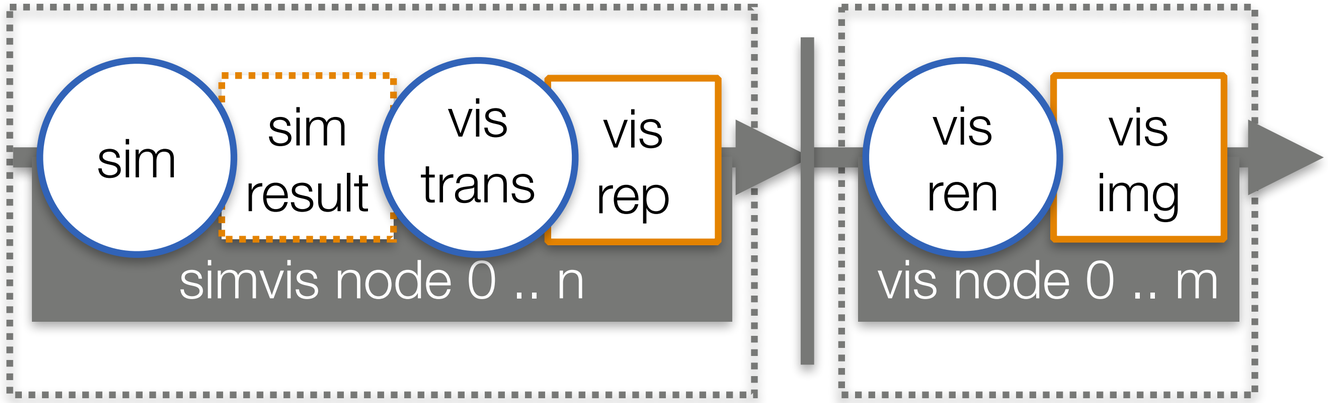


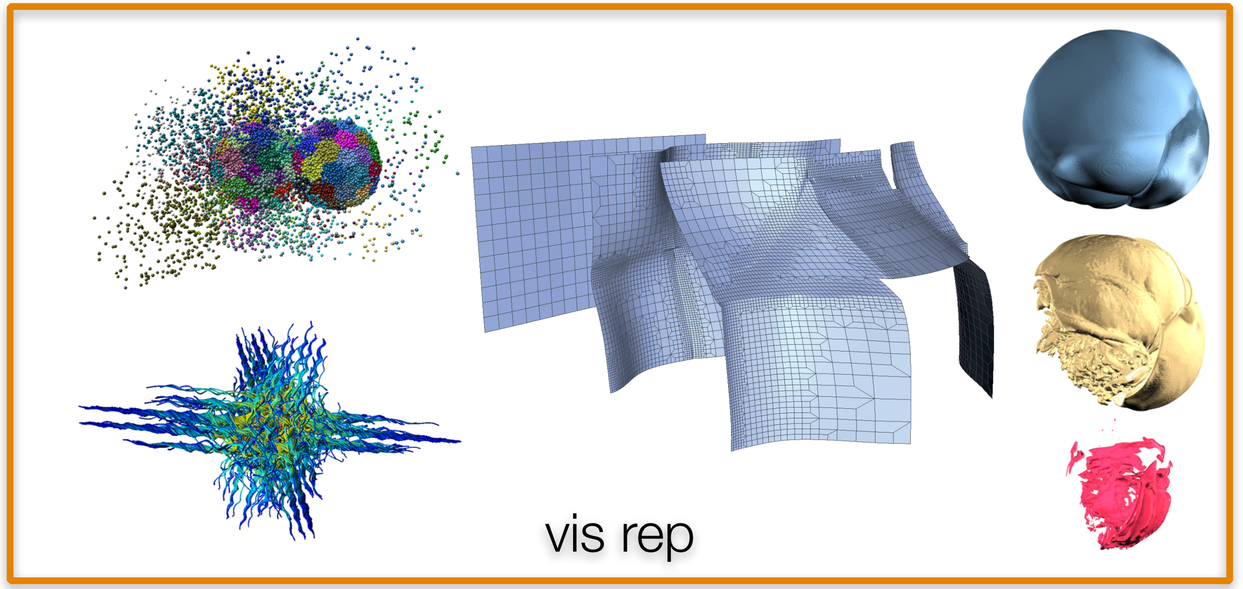
vis img
(raycasting of supernova data set)

the supernova data set is made available by Dr. John M. Blondin at the North Carolina State University through US Department of Energy's SciDAC Institute for Ultrascale Visuaization
(dimension: 432 x 432 x 432, timetsteps: [1295, 1354], data type: float, grid type: uniform, obtained via vidi.cs.ucdavis.edu)

Simulation-Visualization Coupling (4)

- *hybrid coupling*
  - visualization procedure split into two parts
  1. vis trans: generate intermediate visualization representation (vis rep)
     - tight in-situ coupling
     - reduces data & maintains flexibility
  2. vis ren: render image
     - visualization representation is transferred to vis node
     - generate image from intermediate representation
       - based on parameters that can be adjusted (interactively) by a suer
       - e.g., camera position, transfer functions, etc.
  - focus on hybrid coupling in the following

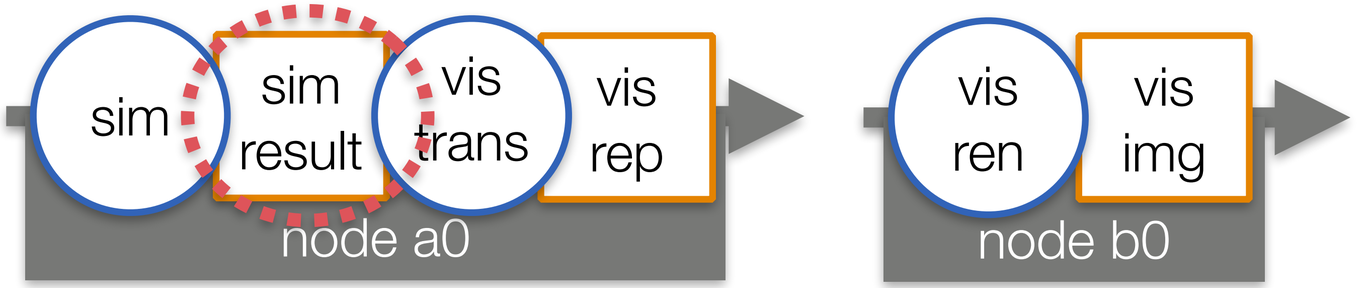


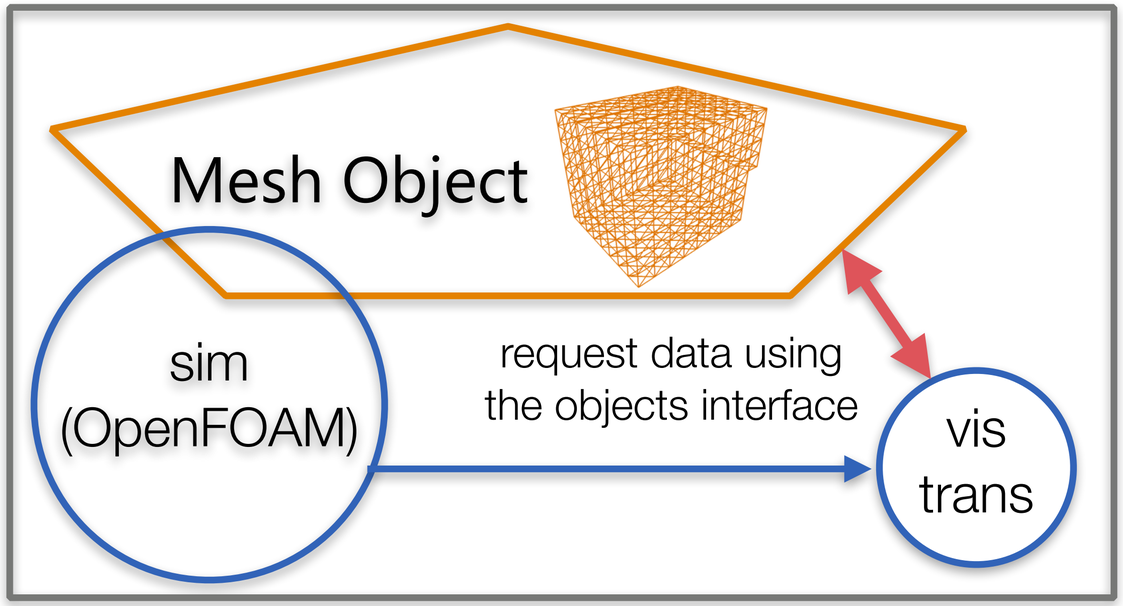


S. Frey, T. Schlömer, S. Grottel, C. Dachsbacher, O. Deussen, and T. Ertl. Loose Capacity-Constrained Representatives for the Qualitative Visual Analysis in Molecular Dynamics. PacificVis, 2011.
S. Frey, F. Sadlo, and T. Ertl. Mesh Generation From Layered Depth Images Using Isosurface Raycasting. ISVC, 2013.

In-Situ Visualization Challenges (1)

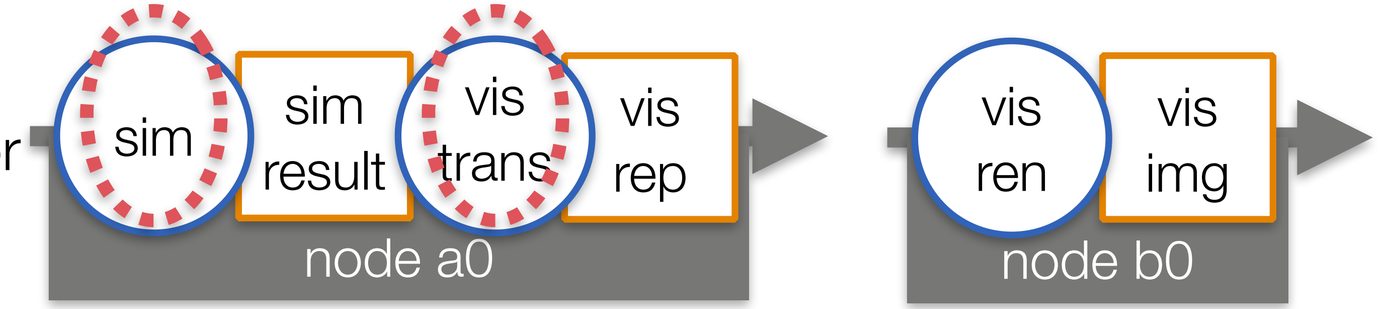
- *simulation-visualization interface*
  - minimize data transfers
  - avoid data conversion wherever possible
  - use optimized data structures from simulation directly for visualization
    - not necessarily the most efficient solution
      - visualization can exhibit very different access characteristics
  - example: raycaster using OpenFOAM
    - simulation and visualization run in the same process
      - visualization embedded as as a library
    - data structures and interface access functions of OpenFOAM are directly used
    - i.e., visualization as a library

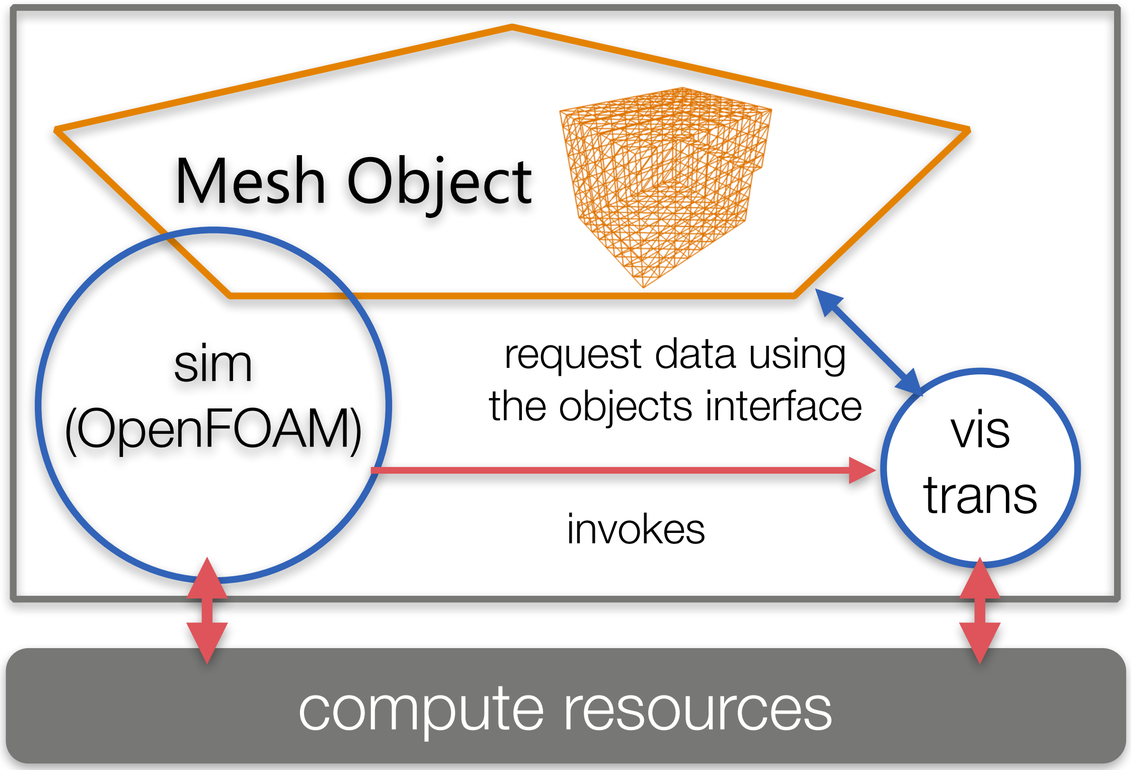




In-Situ Visualization Challenges (2)

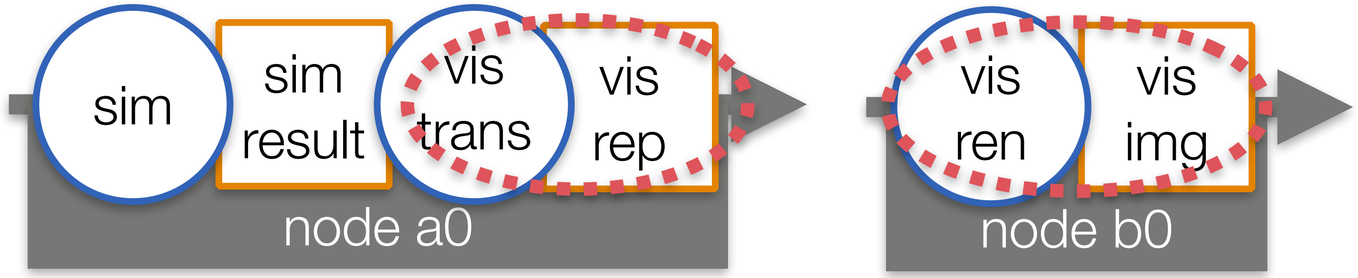
- *resource sharing*
  - visualization competes with simulation for compute performance
  - scheduling
    - vis trans can be deferred
    - utilize open slots due to load-imbalance
      - e.g., determined by simulation scheduler
  - cost control
    - make vis trans flexible
      - quality-resource consumption tradeoff
    - determine acceptable cost for vis trans
  - example
    - simulation invokes visualization
    - resolution of generated vis rep used as parameter for cost vs quality

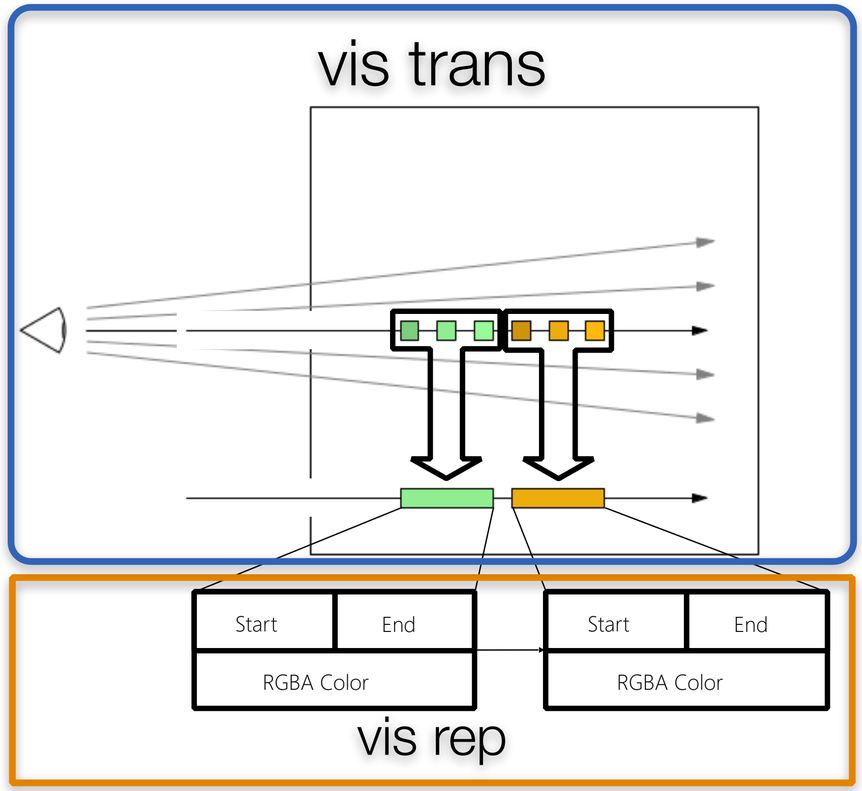




In-Situ Visualization Challenges (3)

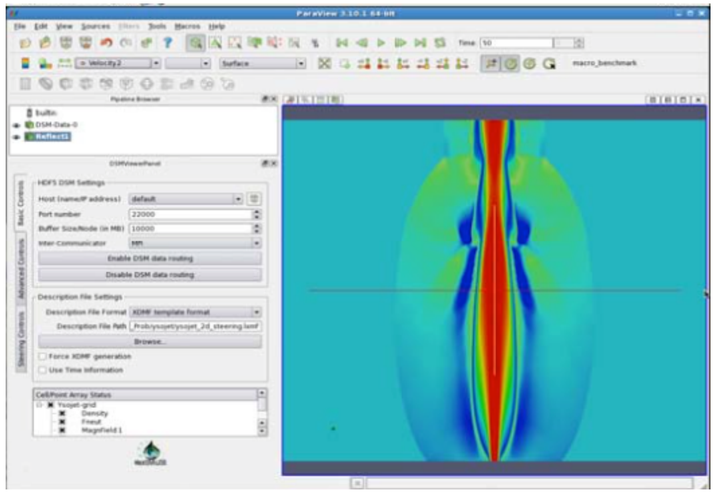
- *reduced/intermediate visualization representation*
  1. should be of small size, …
  2. fast and efficient to generate, …
  3. … , and still represent information of interest
     - with some exploration capabilities preserved
  - two main development approaches
    - start from standard visualization approach and reduce/ limit degrees of freedom
    - tailor toward specific domain research question
  - example (VDI, more detail in the following)
    - fix mapping of color to values (transfer function)
    - optimize for a certain view, but generally allow arbitrary camera movement (impact on quality)
  - focus on this in the following



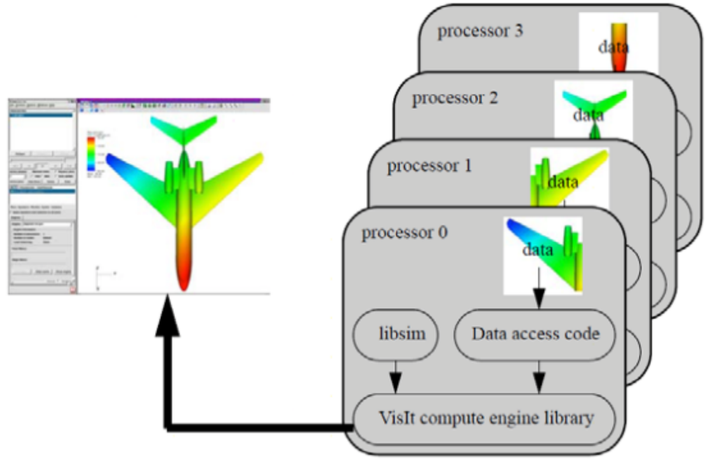


State of the Art in In-Situ Visualization (1)

- *frameworks implementing loose and tight coupling*
  - Paraview ICARUS (Initialize Compute Analyze Render Update Steer)
    - loosely coupled and push-driven
      - HDF5 file in shared memory
      - no modification of simulation codes
  - ParaView Co-processing
    - simulation communicates with outside visualization tool that generates images
      - co-processing library is embedded in simulation (tight coupling)
      - graphical configuration tools for co-processing configuration
  - VisIt (libsim)
    - user needs to implement access functions required by VisIt in libsim (tight coupling)
      - simulation-specific
    - provides interfaces to VisIt



ICARUS

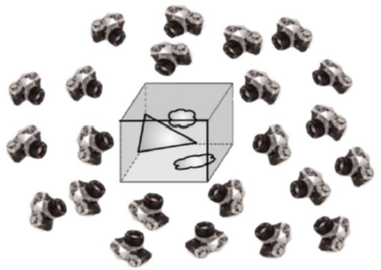


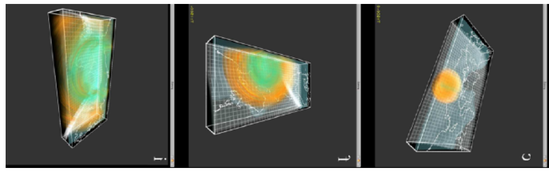
VisIt

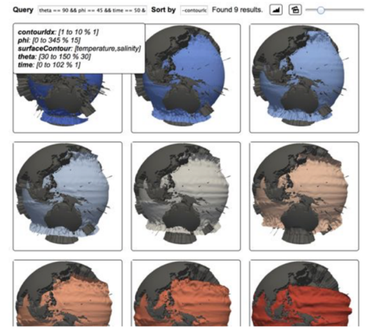
screenshots from Rivia et al.

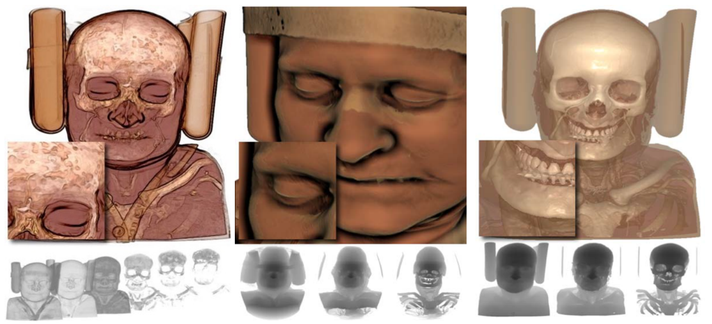
State of the Art in In-Situ Visualization (2)

- *reduced representations for volume rendering*
  - interactive Viewing [Kageyama and Yamada]
    - *visrep:* record videos from different views
    - *vis:* interactive view of many movies (camera selection)
  - image data bases [Ahrens et al.]
    - *visrep*: store a set of output images directly from the simulation into an image database
    - vis: content querying and creation of new images by composing of individual visualization objects
  - proxy images [Tikhonova et al.]
    - *visrep*: collection of proxy images containing different information (depth, different view, etc.)
    - vis: mix and match images





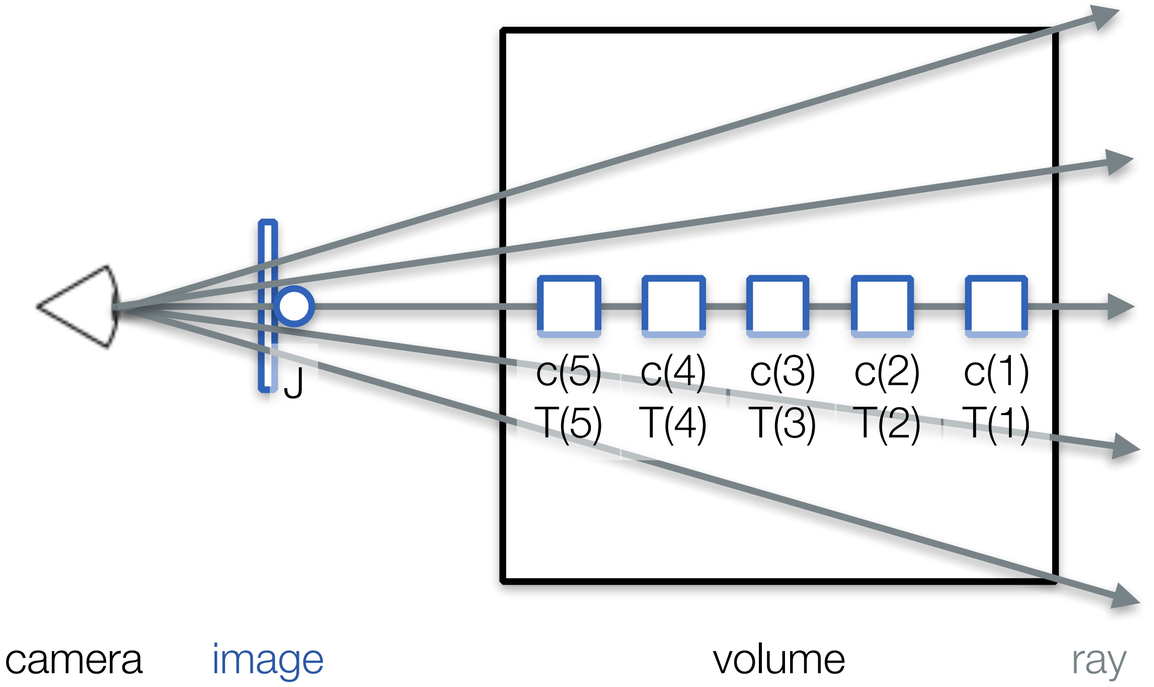




[Tikhonova et al.] Tikhonova, A., Correa, C. D., & Ma, K.-L. (2010). Visualization by Proxy: A Novel Framework for Deferred Interaction with Volume Data. IEEE Transaction on Visualization and Computer Graphics, 16(6), 1551–1559.
[Ahrens et al.] Ahrens, J., Jourdain, S., O'leary, P., Pratchett, J., Rogers, D. H., & Petersen, M. (2014). An Image based Approach to Extreme Scale In Situ Visualization. SC '14 Proceedings, 424–434.
[Kageyama and Yamada] Kageyama, A., & Yamada, T. (n.d.). An approach to exascale visualization: Interactive viewing of in-situ visualization. Computer Physics Communications, 79-85.

Volumetric Depth Images

- volumetric depth images (VDIs)
  - directly based on volumetric raycasting
    - standard technique for volume visualization
  - intermediate representation from (partial) volumetric raycasting
- volumetric raycasting
  - send one ray through each pixel
  - sample along ray
  - accumulate color (c) and opacity / transparency (T)

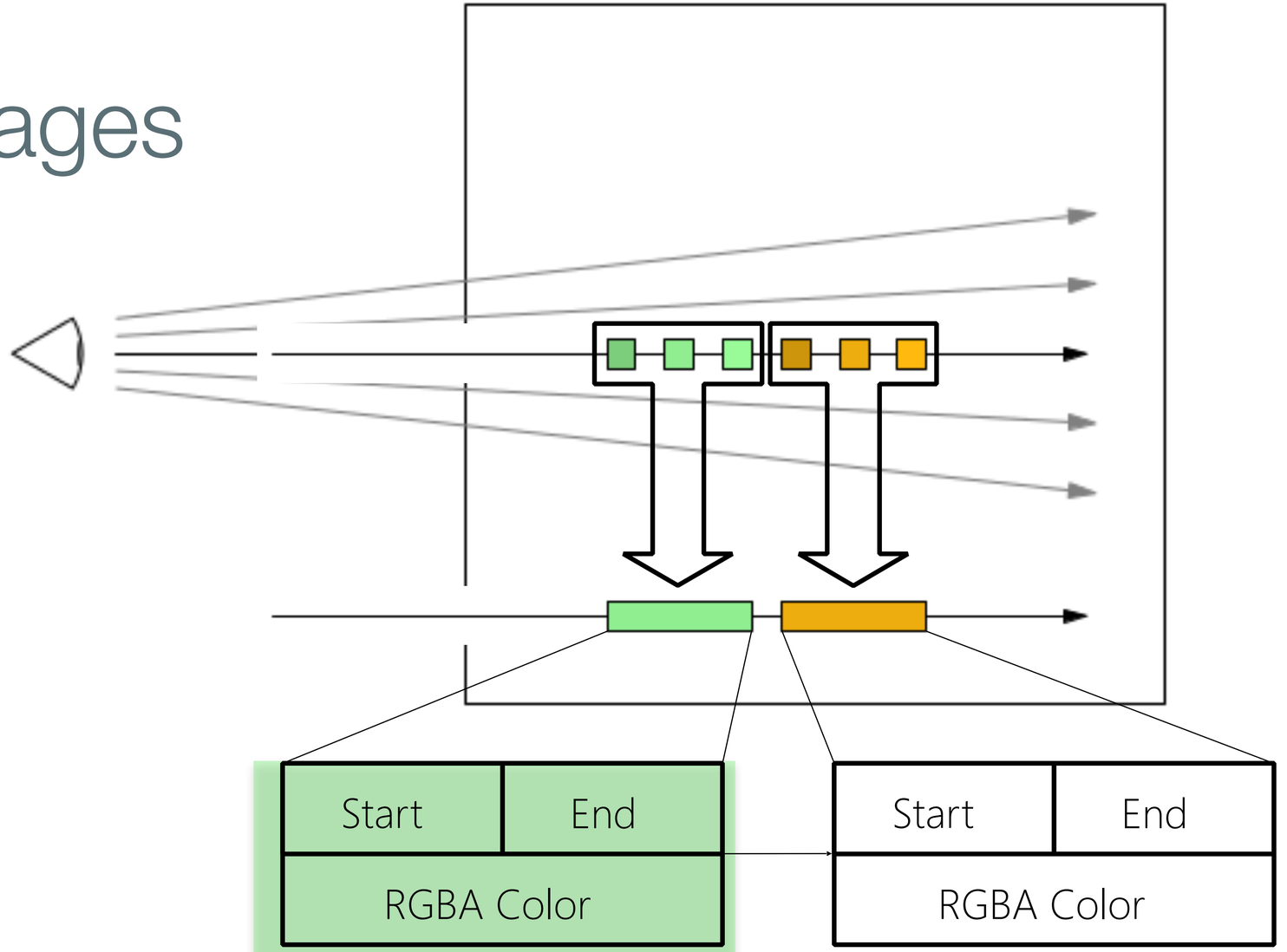


$$J(D) := \sum_{i=1}^{N} c(i) \prod_{j=i+1}^{N} T(j),$$

S. Frey, F. Sadlo, and T. Ertl. Explorable volumetric depth images from raycasting, SIBGRAPI, 2013.

Volumetric Depth Images

- subsequent samples can be arbitrarily grouped into segments and composited
  - no loss in quality
- Enabling “hierarchical” representations
  - segments represented by RGBA + 2 depth values

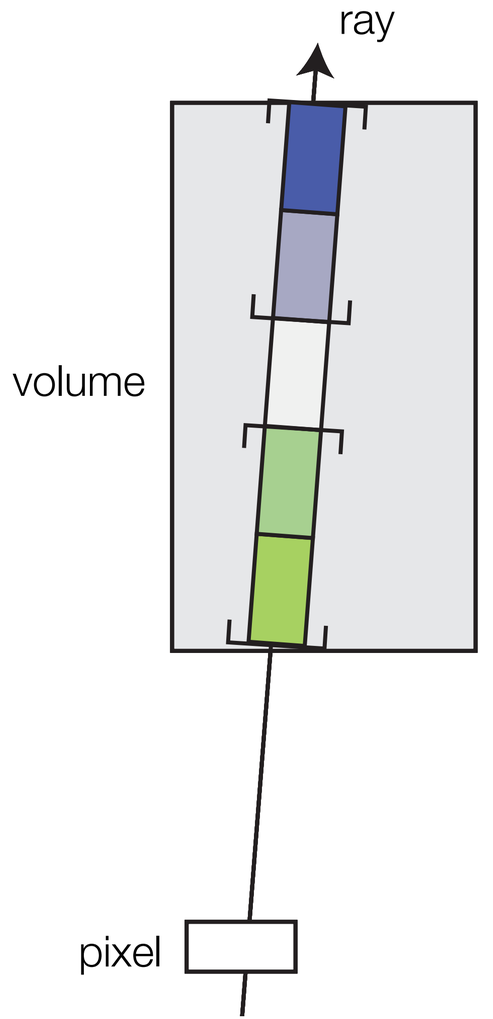


$$J(D) := \sum_{i=1}^{N} c(i) \prod_{j=i+1}^{N} T(j) \Rightarrow J(D) = \sum_{p=1}^{P} \left(\left(\sum_{i=s_b(p)}^{s_f(p)} c(i) \prod_{j=i+1}^{s_f(p)} T(j) \right) \prod_{k=s_f(p)+1}^{N} T(k) \right)$$

VDI Generation

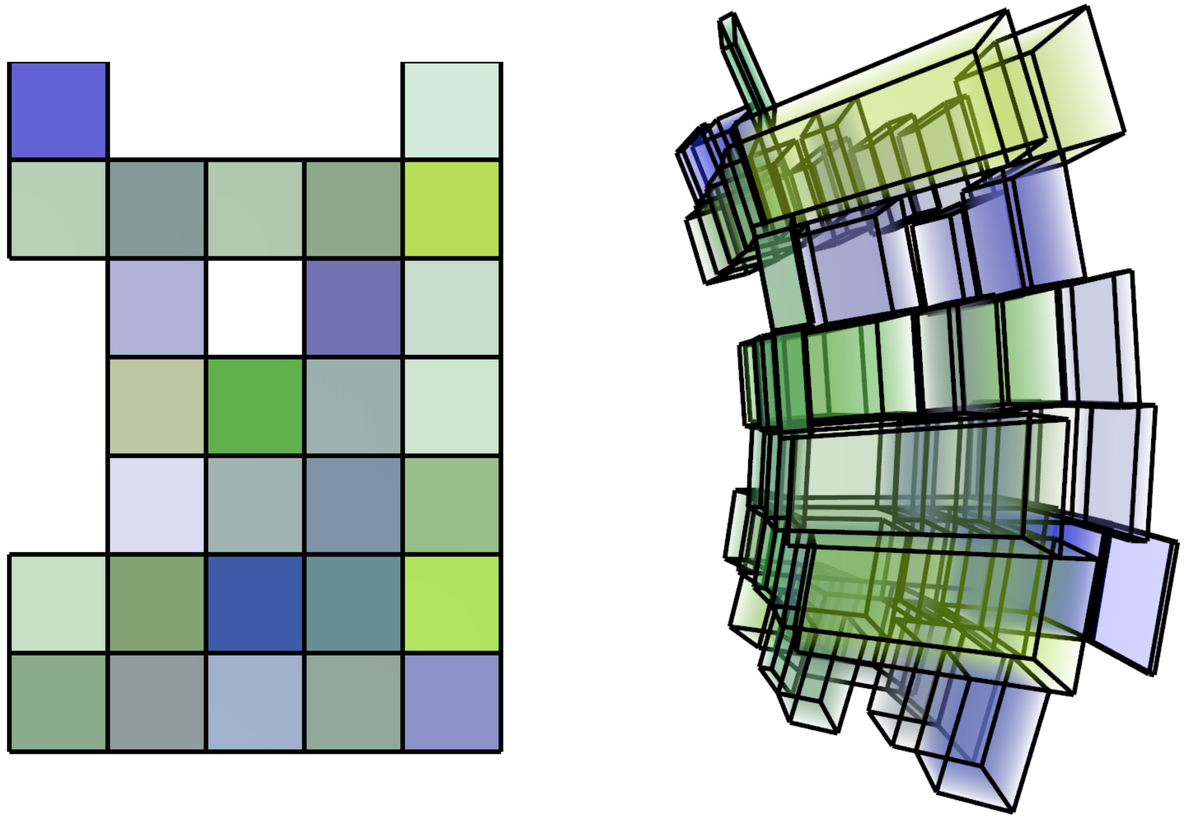
- modified front-to-back raycasting procedure
- main adjustment: color and opacity are not composited over the whole ray but only within supersegments
- a raycaster segment is merged into the current segment if ...
- ... the adjusted color and opacity difference is below the user-provided sensitivity parameter γ
- ... and the sample does not represent empty space

$$\gamma > |\underbrace{(c_{s_f}^{i-1}, \alpha_{s_f}^{i-1})}_{\text{new segment RGBA}} - \underbrace{(\hat{\alpha}(s_f, i)C(i), \hat{\alpha}(s_f, i))}_{\text{current supersegment RGBA}}|$$



Rendering Volumetric Depth Images

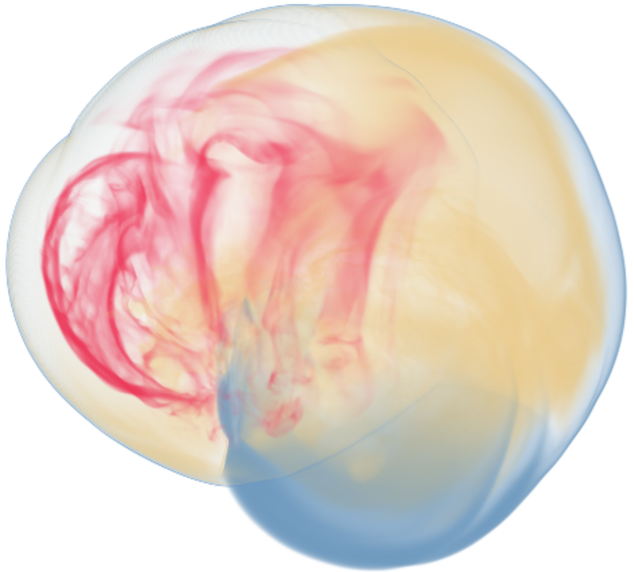
- create frustums: geometric representation of segments
  - using RGBA+depth information
  - camera configuration of original view
- rendered efficiently using OpenGL & shaders
  - alpha-blending with color and opacity from the segments
    - contribution of each frustum adjusted w.r.t. length of view ray passing through it
    - efficient depth sorting of frustums by exploiting knowledge about their creation



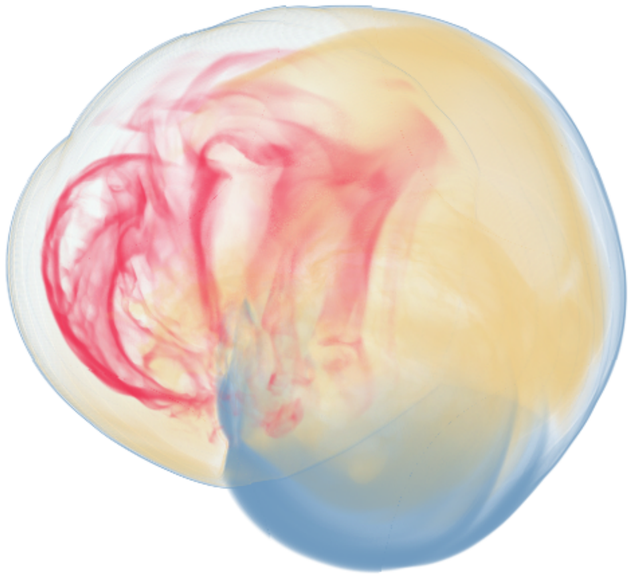
(low resolution) render representation
(from front and 90° rotation)

Volumetric Depth Images

- VDI exactly matches raycasting from initial view
  - for small deviations minor visual difference ....
  - ... that increases with larger view angles
- VDI rendering performance approximately one order of magnitude faster
  - 34ms vs 275ms
- VDI generation fast
  - few milliseconds overhead to standard raycasting

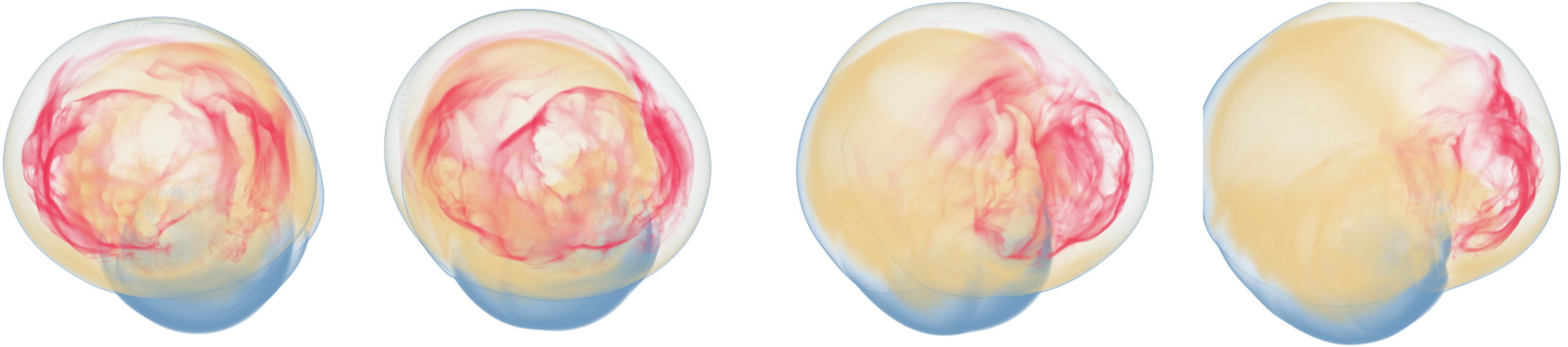


raycasting 80M cells in 275ms per frame

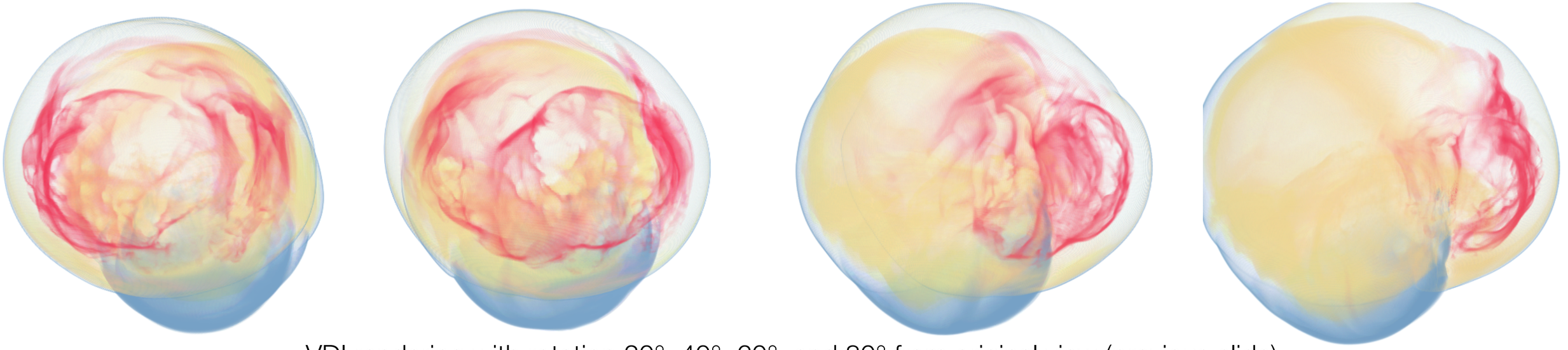


VDI rendering of 1M frustums in 34ms per frame (original view)

Volumetric Depth Images



raycasting



VDI rendering with rotation 20°, 40°, 60°, and 80° from original view (previous slide)

Space-Time Volumetric Depth

- so far: samples only accumulated along rays
- space-time extension: accumulate segments
  - across rays (in image space) and across time
  - →store depth values and color more efficiently
    - (spend more time and accept some error for a further reduction in data size)
- procedure
  - determine local neighborhood in space and time
    - segments uniformly arranged (simple depth comparison)
  - defines connectivity graph of all segments
  - cluster regions of color
    - quantize depth values and color values
    - additional delta encoding of depth values
  - lossless compression of results

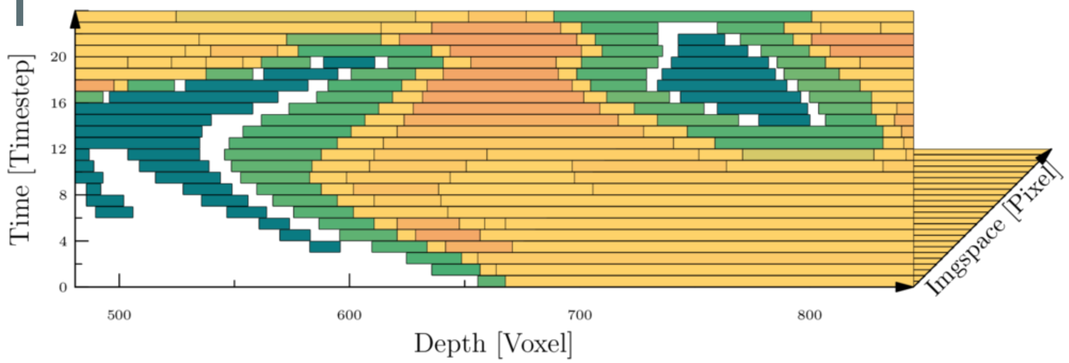
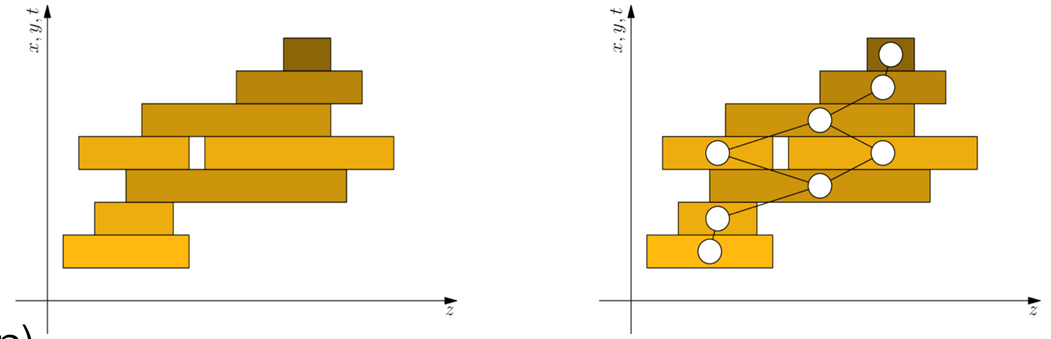
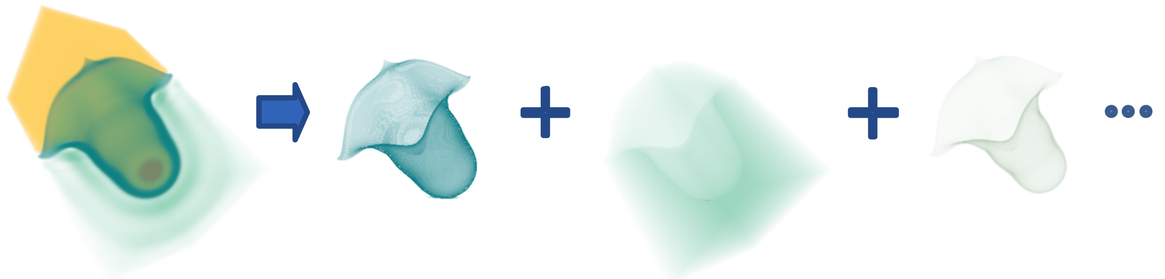


illustration of extracted segments



graph of segments

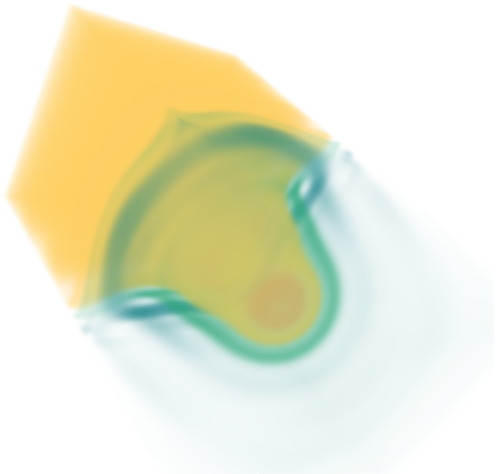


region clusters

O. Fernandes, S. Frey, F. Sadlo, and T. Ertl. Space-time volumetric depth images for in-situ-visualization. LDAV, 2014.

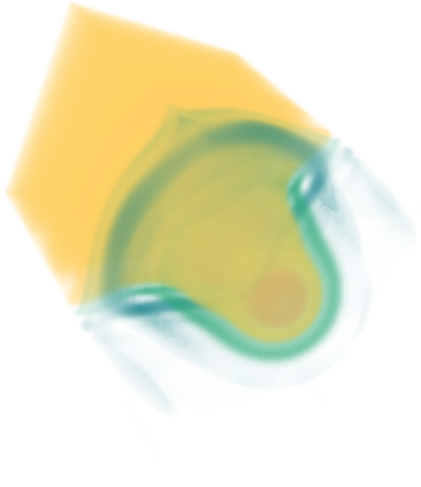
Space-Time VDI Results

- significant reductions can be achieved over standard VDIs
- how much depends on data set and parameters
  - quantization of depth values
  - color threshold

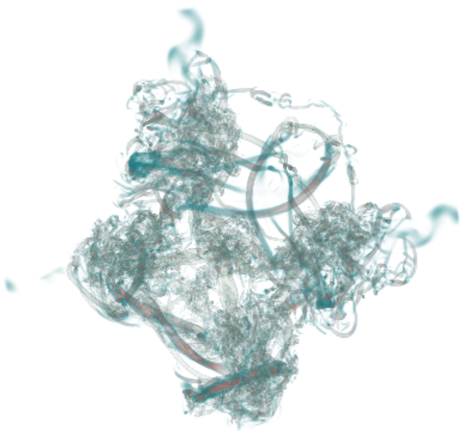


Original size
~85.0 MB
(raycasted)

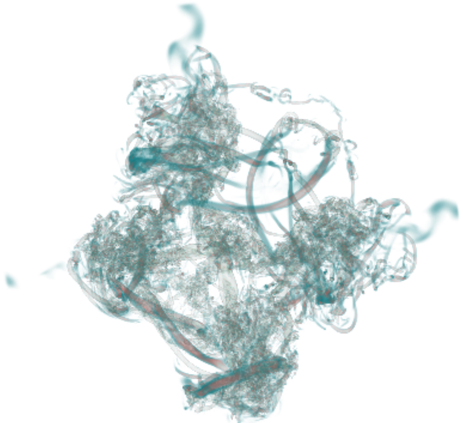
Original VDI
~40.0 MB
(lossless enc.)



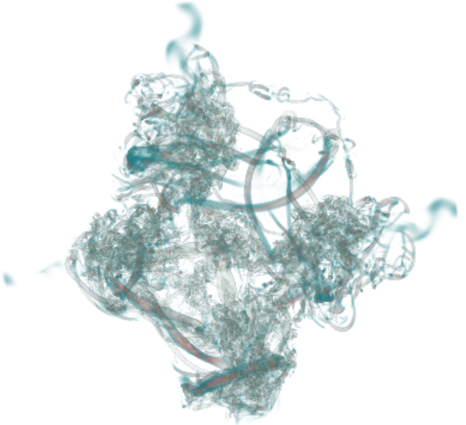
Space-Time VDI
~5.6 MB
(lossless enc.)



Original size
~2130 MB
(raycasted)



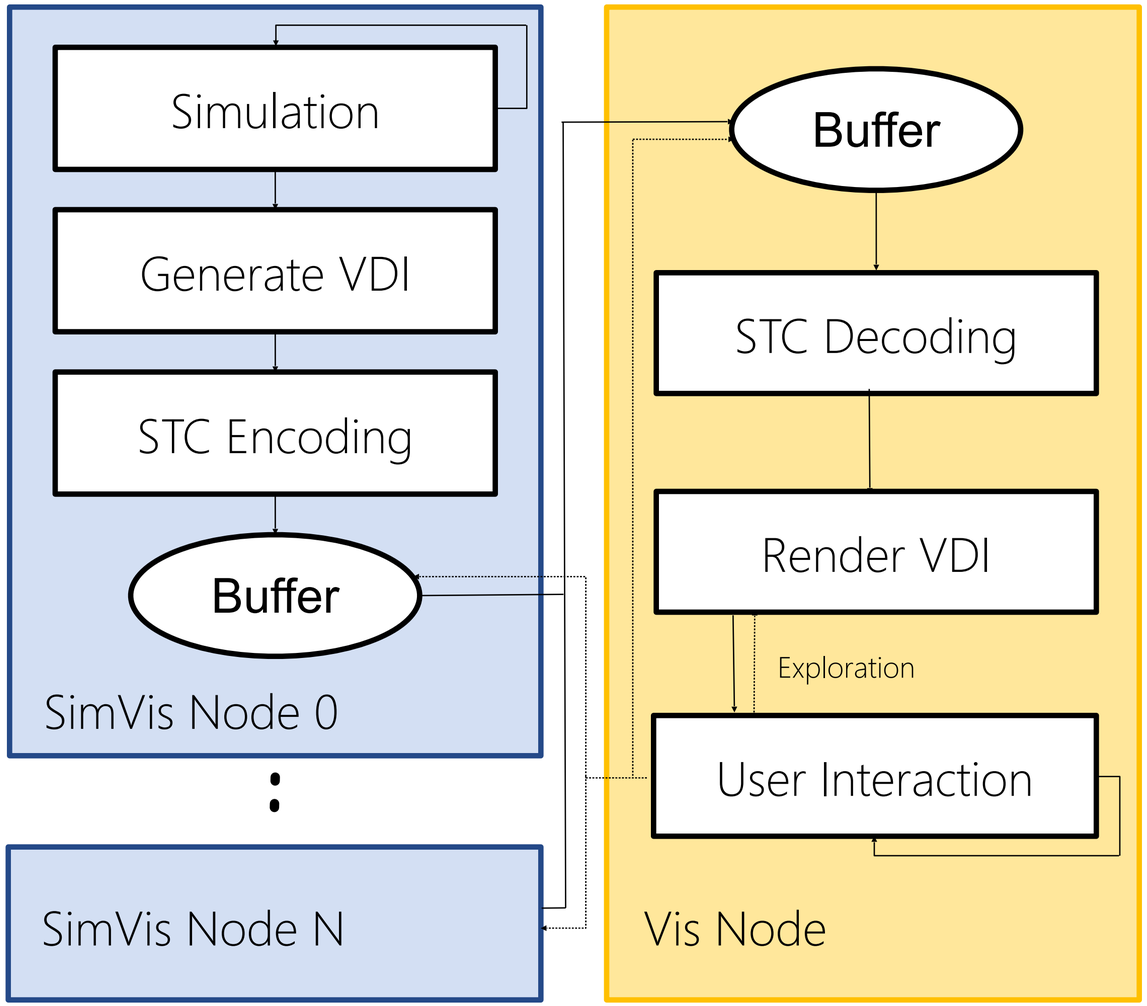
Original VDI
~375 MB
(lossless enc.)



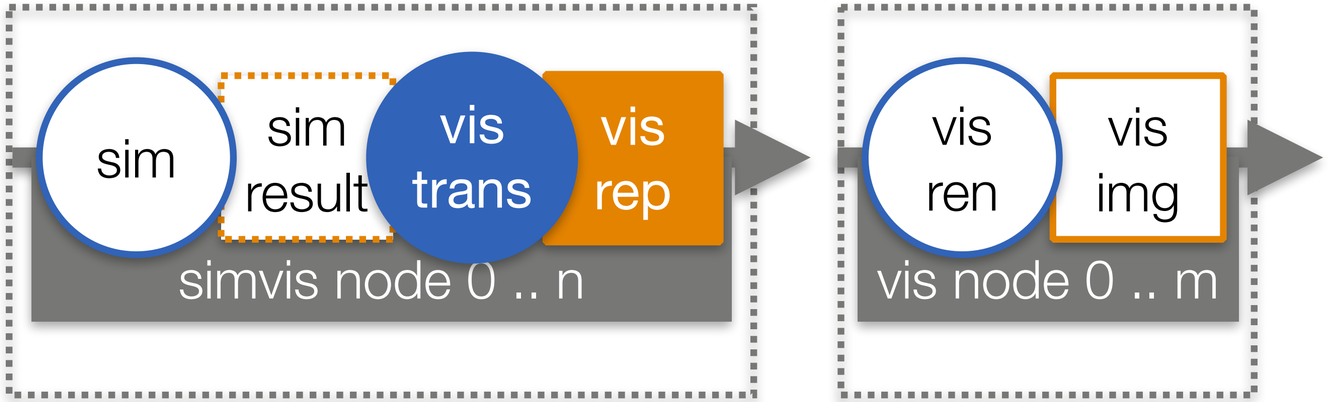
Space-Time VDI
~247 MB
(lossless enc.)

Overall Architecture

- simulation-visualization node
  - VDI is generated as soon as simulation generates a time step
  - space-time coding reduces data size
    - continuously updated with new time steps
- visualization node
  - pipeline is executed in reverse order
  - allows for interactive exploration of each time step
  - … and temporal navigation
  - sliding window spans certain time interval
    - old time steps might have to be discarded if certain limited buffer size is reached



Conclusion



- in-situ visualization aims to
  - … minimize the cost for transfer and storage
  - … with a low computational overhead
  - … keep the data of interest
  - … maintain the ability of a user for a posteriori exploration
  - … be available with minimum delay
- hybrid coupling in-situ architecture can achieve this
- … but requires the development of reduced visualization representations
  - research-question and application-specific
  - discussed some of our work in the area
- ongoing challenges
  - simulation-visualization coupling and resource sharing considering software architecture and runtime characteristics of simulation
  - interdisciplinary choice and development of visualization representation