Hands-on Seminar of Remote Visualization System PBVR

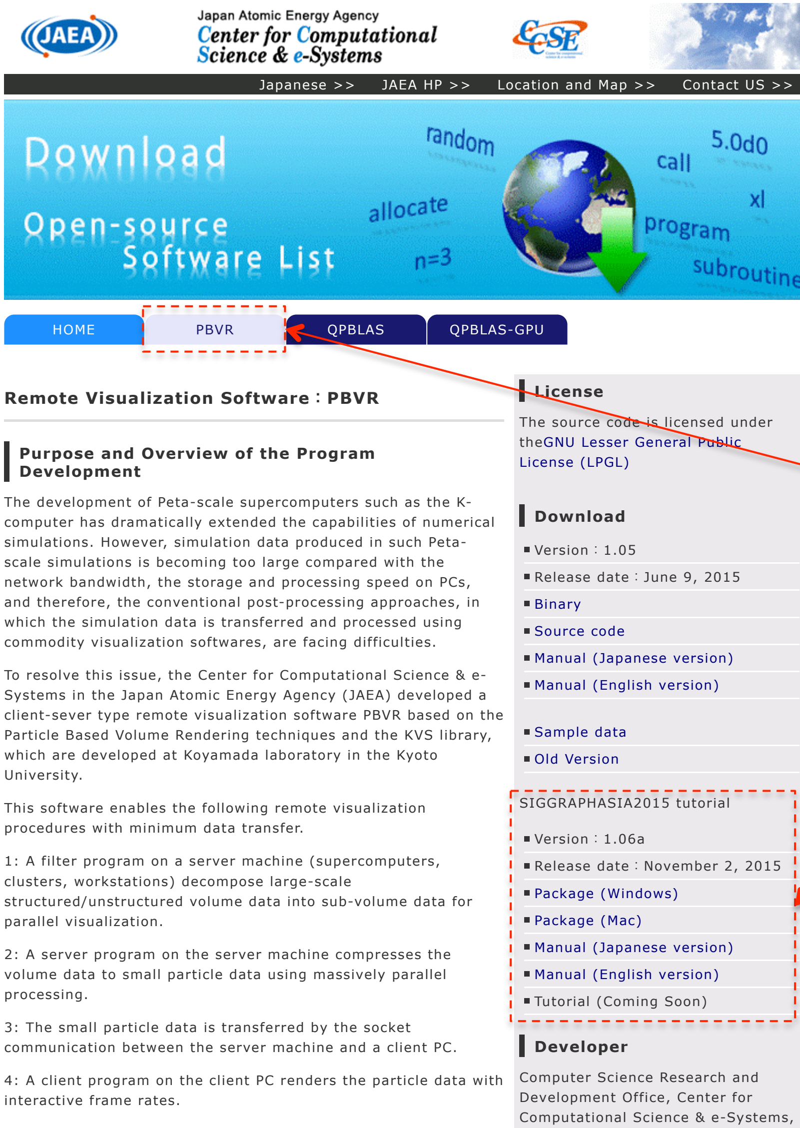
Takuma Kawamura

Japan Atomic Energy Agency, Chiba, Japan

Siggraph Asia 2015 Symposium On Visualization In High Performance Computing, Kobe Convention Center, Japan

Download

http://ccse.jaea.go.jp/ja/download/software_eng.html



Japan Atomic Energy Agency
Center for Computational Science & e-Systems

Japanese >> JAEA HP >> Location and Map >> Contact US >>

Download
Open-source Software List

HOME | PBVR | QPBLAS | QPBLAS-GPU

Remote Visualization Software：PBVR

Purpose and Overview of the Program Development

The development of Peta-scale supercomputers such as the K-computer has dramatically extended the capabilities of numerical simulations. However, simulation data produced in such Peta-scale simulations is becoming too large compared with the network bandwidth, the storage and processing speed on PCs, and therefore, the conventional post-processing approaches, in which the simulation data is transferred and processed using commodity visualization softwares, are facing difficulties.

To resolve this issue, the Center for Computational Science & e-Systems in the Japan Atomic Energy Agency (JAEA) developed a client-sever type remote visualization software PBVR based on the Particle Based Volume Rendering techniques and the KVS library, which are developed at Koyamada laboratory in the Kyoto University.

This software enables the following remote visualization procedures with minimum data transfer.

1: A filter program on a server machine (supercomputers, clusters, workstations) decompose large-scale structured/unstructured volume data into sub-volume data for parallel visualization.

2: A server program on the server machine compresses the volume data to small particle data using massively parallel processing.

3: The small particle data is transferred by the socket communication between the server machine and a client PC.

4: A client program on the client PC renders the particle data with interactive frame rates.

License

The source code is licensed under theGNU Lesser General Public License (LPGL)

Download

- Version：1.05
- Release date：June 9, 2015
- Binary
- Source code
- Manual (Japanese version)
- Manual (English version)
- Sample data
- Old Version

SIGGRAPHASIA2015 tutorial

- Version：1.06a
- Release date：November 2, 2015
- Package (Windows)
- Package (Mac)
- Manual (Japanese version)
- Manual (English version)
- Tutorial (Coming Soon)

Developer

Computer Science Research and Development Office, Center for Computational Science & e-Systems,

or search keywords

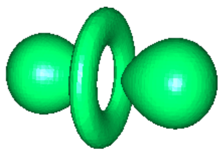
“ **ccse download pbvr** ”

Click “PBVR” tab

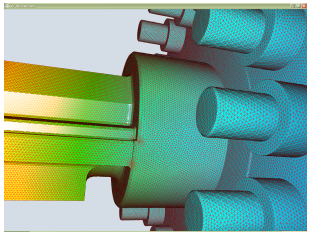
Get hands-on package of version 1.06a

- Overview of PBVR
  – Background
  – Client-server visualization based on PBVR
- Brief demonstration of PBVR with client server mode
  – Remote visualization of large data on K computer
- Hands-on with PBVR
  – Install prebuilt binary of PBVR
  – Run PBVR system with standalone mode
  – Become familiar with viewer window
  – Image quality control
  – Designing transfer functions
  – Saving results

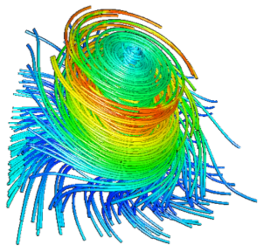
Visualization of Large-Scale Data



Hydrogen charge density

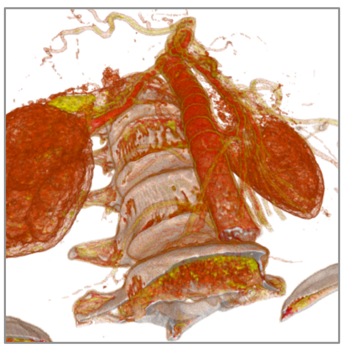


Pump structure

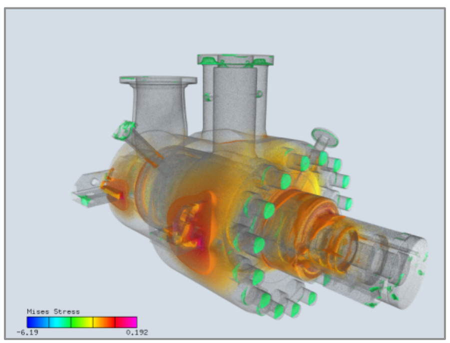


Tornado flow

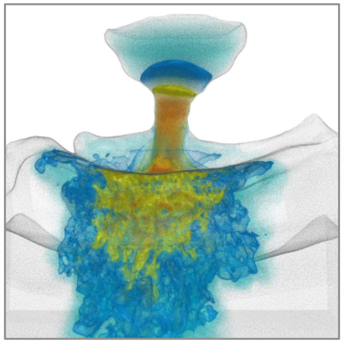
Volume Rendering



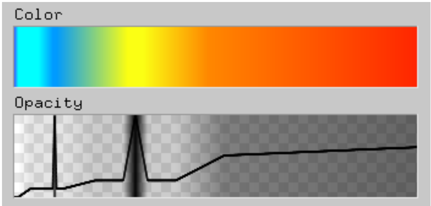
Breast CT scan



Pump structure



Oral air flow



Transfer function

Horizontal : physical value

Vertical : color and opacity

Various Visualization for Analysis

- Iso-surface, Boundary surface, Stream line, etc......
- Volume rendering
  - Efficient technique on various fields
- Interactive operation of visualization parameter (Threshold value, color, viewing position, , etc......)
  - Discovering important feature

Increasing Simulation Scale

- Large-scale data (tera, peta, and larger)
- Low interactivity inhibits analysis

Interactive visualization for remote large-scale data (few second scale)

Background: Remote Visualization of Large-scale Data

Traditional Visualization

- Transferring original data to PC
- Impossible to transfer large data

Data Compression[1]

- Data redundancy of time variance or frequency characteristics
- Difficult to select compression method suitable to data type

Client-Server Type Visualization[2]

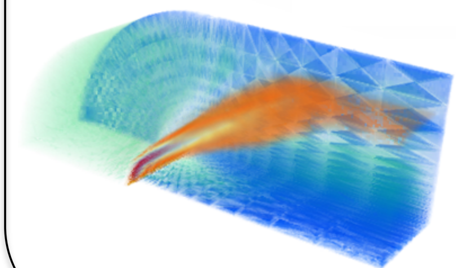
- Transferring visualization primitive (polygon)
- High interactivity of viewing position
- Increasing rendering primitive in extreme-scale visualization

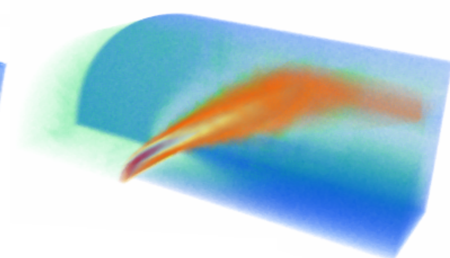
Parallel Volume Rendering[3,4,5]

- Suitable for extreme-scale visualization on parallel environment
- Strong scaling is limited by image composition phase (alpha blending)
- Low interactivity of viewing position

Visibility Order for Alpha Blending

- Semi-transparent primitives (polygon, sampling point, etc…) are composited from the order of viewing position


Sorting miss


Sorting correct

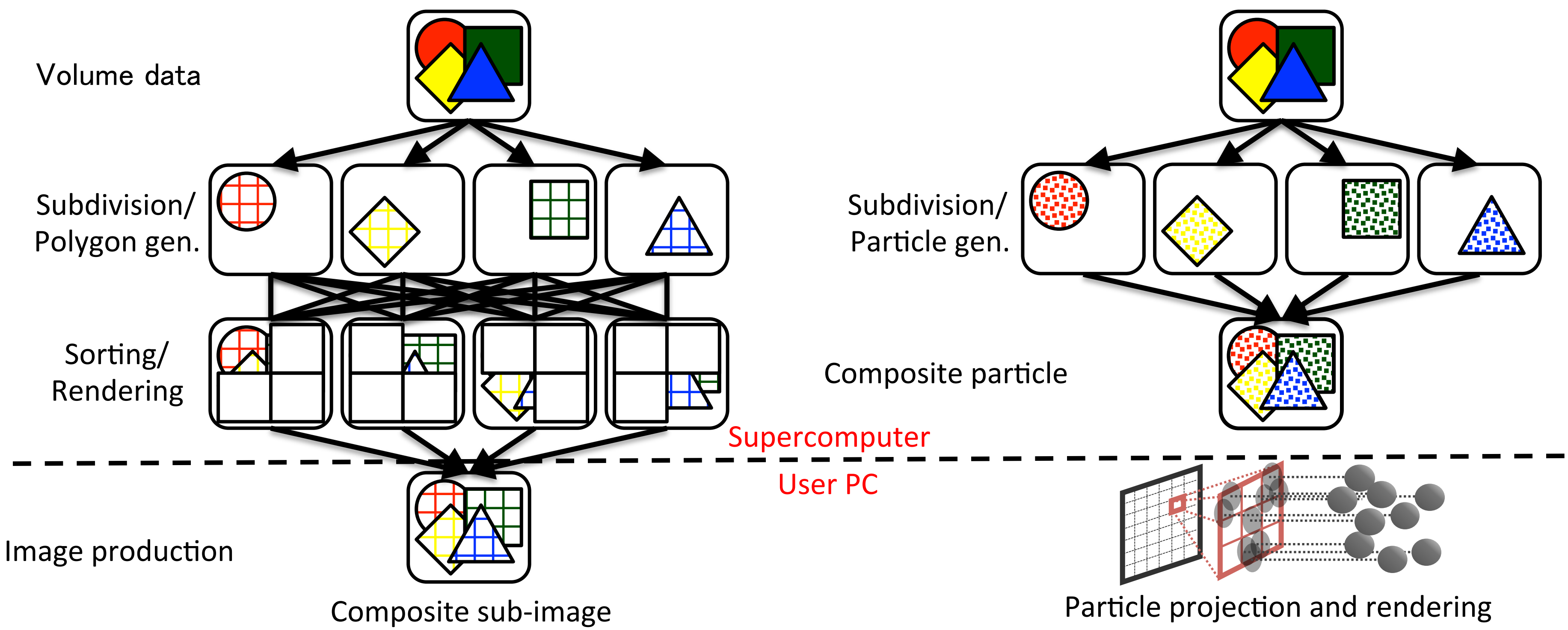
1. H. Ohtani, K. Hagita, A. M. Ito, et al. "Time-Order Kinetic Irreversible Compression Scheme for ......", IEEE VIS 2013
2. Ensight, Visit, ParaView, etc…
3. M.Howison, E.W.Bethel, et al. "Hybrid parallelism for volume rendering on large ......", IEEE Trans. VCG 2012
4. T.Peterka, H.Yu, R.Ross, K.L.Ma, "Parallel Volume Rendering on the IBM Blue Gene/P", EGPGV 2008
5. K. Ono, J. Nonaka, "Design of cooperative visualization environment with intensive data ......", Ultra Vis 2008

Polygon-based Volume Rendering

- Large polygon data (~=volume data)
- Image generation for sub-region
- Polygon sorting requires heavy comm.
  → Limited strong scaling
- Recalculation for interactive process

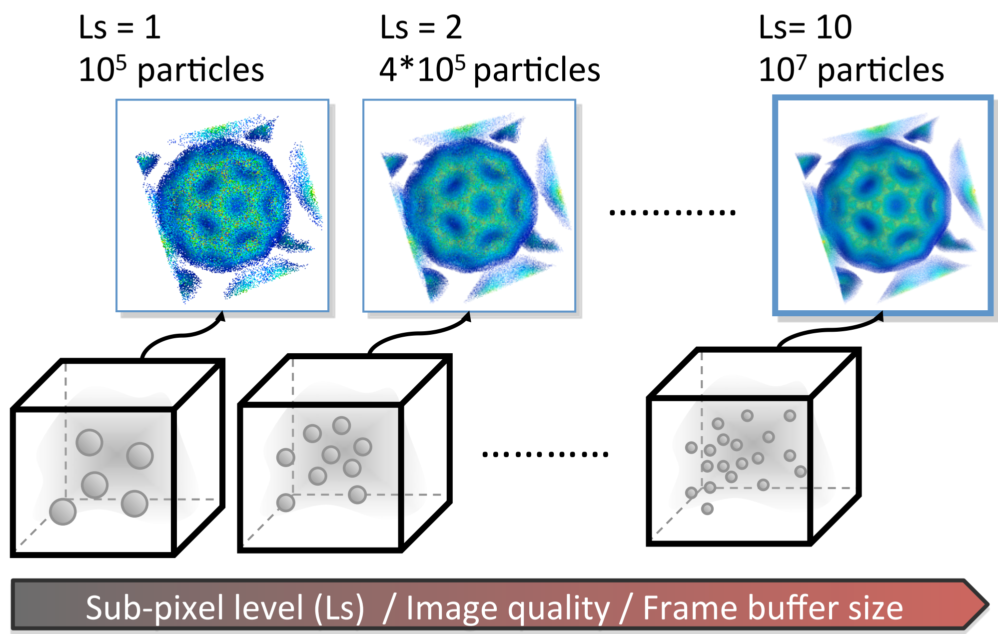
Particle-based Volume Rendering

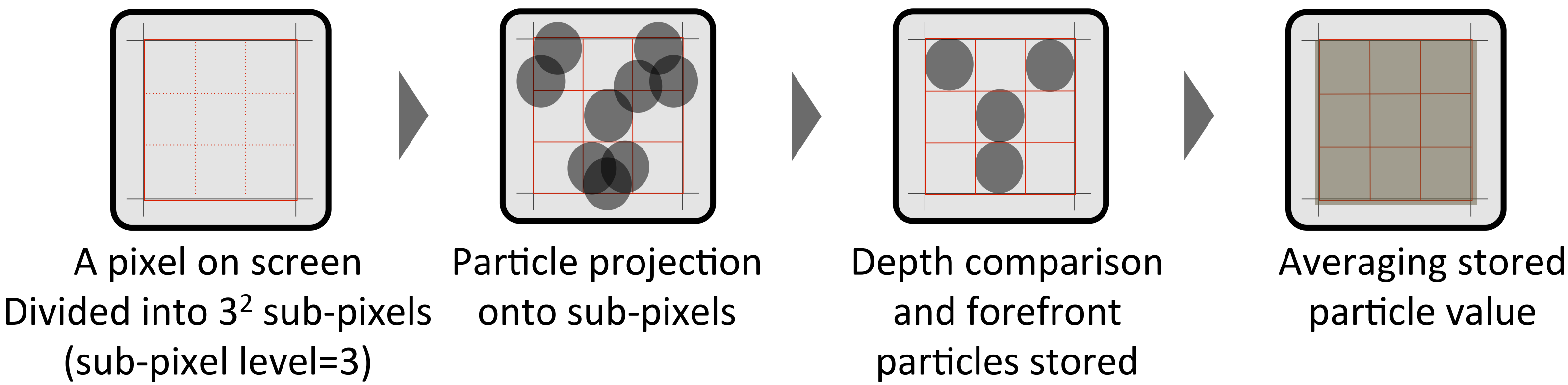
- Small particle data
- Particle generation for sub-region
- Particle composition without sorting
  → Suitable for parallel processing
- Interactive processing on PC



Mechanism of PBVR[1,2]

- Particle density calculated from opacity
- Particle generation by Monte-Carlo sampling
  - Metropolis /Rejection /Uniform sampling
- Pixel division (sub-pixel)
  - Particle diameter equals to sub-pixel length
  - Sub-pixel expression by frame buffer
- Fast Z-buffer algorithm for depth comparison



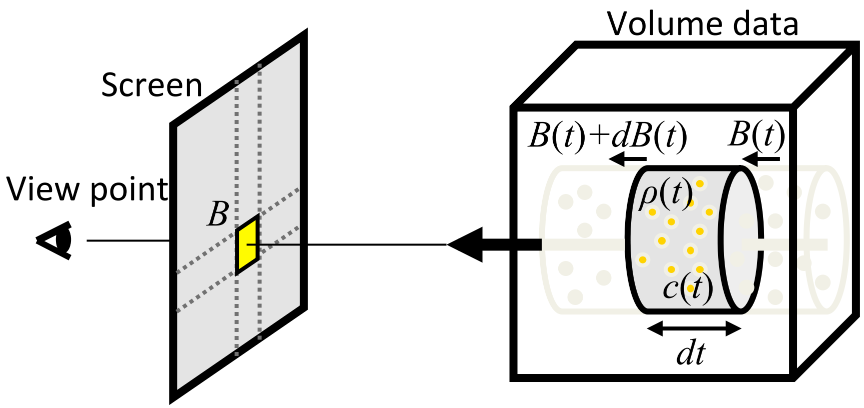


1. N. Sakamoto, J. Nonaka, K. Koyamada, S. Tanaka, "Particle-based Volume Rendering", APVIS 2007
2. T. Kawamura, N. Sakamoto, K. Koyamada,"Level-of-detail Rendering of a Large-scale Irregular Volume .... ", JCST 2010

Optical Model of Volume Rendering

Density Emitter Model[1]

- ❖ Particle distribution in volume data
- ❖ Emission and absorption of light
- ❖ Calculation of brightness



t : distance from view point r : particle radius

$B(t)$: brightness $\rho(t)$: density $c(t)$: brightness per particle

$$B(t_0) = \int_{t_n}^{t_0} c(t) \times \pi r^2 \rho(t) \times \exp\left(-\int_{t}^{t_0} \pi r^2 \rho(\lambda)\, d\lambda\right) dt$$

Brightness equation

Alpha Blending

- ❖ Discretization of brightness equation
- ❖ Opacity defined by density

$$\alpha_k := 1 - \int_{t_k}^{t_{k-1}} \exp\left(-\pi r^2 \rho(t)\right) dt$$

⇩

- ❖ Shielding rate of light emitted from a particle to view point
  - ➢ Light reaching rate follows Poisson distribution

1. P. Williams, N. Max, “A volume density optical model”, VVS 1992

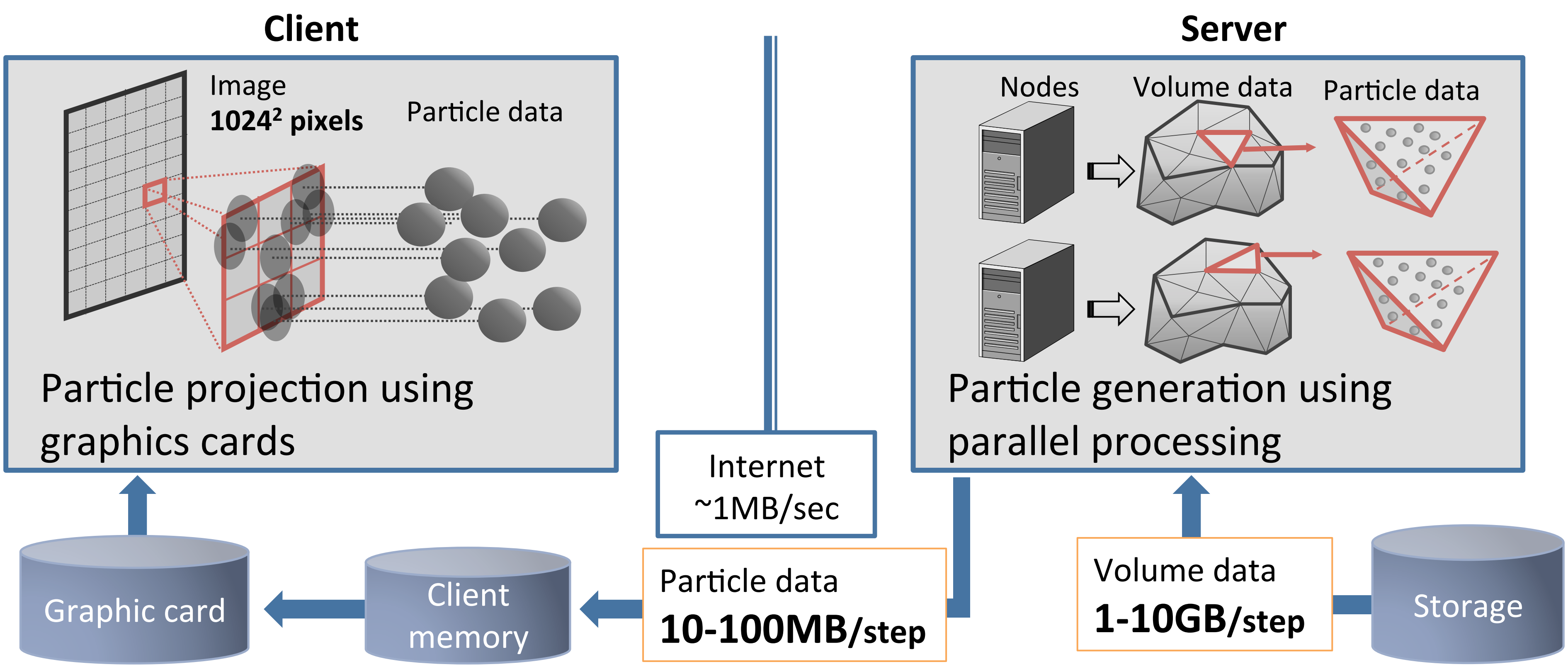
Remote Visualization System using PBVR[1]

Client-server distributed processing model

1. Particle data size smaller than volume data size
2. Sort-less algorithm and Monte-Carlo sampling
3. Rendering process using Graphic card
4. Image quality controlled by number of particles

⇨

1. Client-Server
2. Highly parallelized
3. Interactive viewing position
4. Flexible LoD control



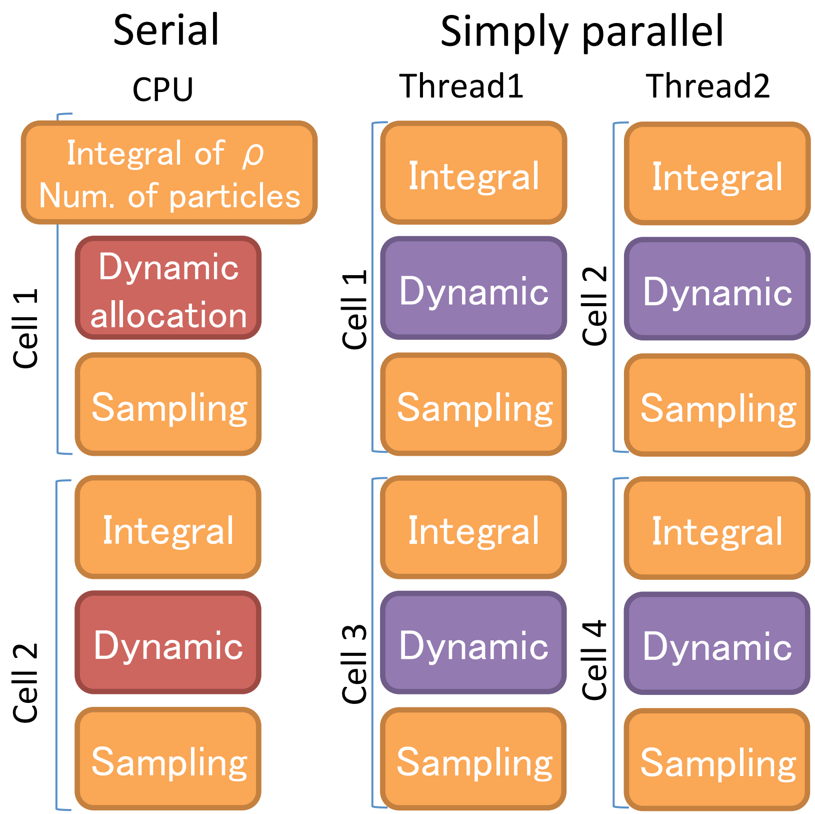
1. T. Kawamura, Y. Idomura, et al. "Remote Visualization for Large-scale Simulation using PBVR", SC 2012

Parallel Particle Generation（GPGPU）

- Small particle data size
- No sorting for particle

⇩

Thread parallel for cell



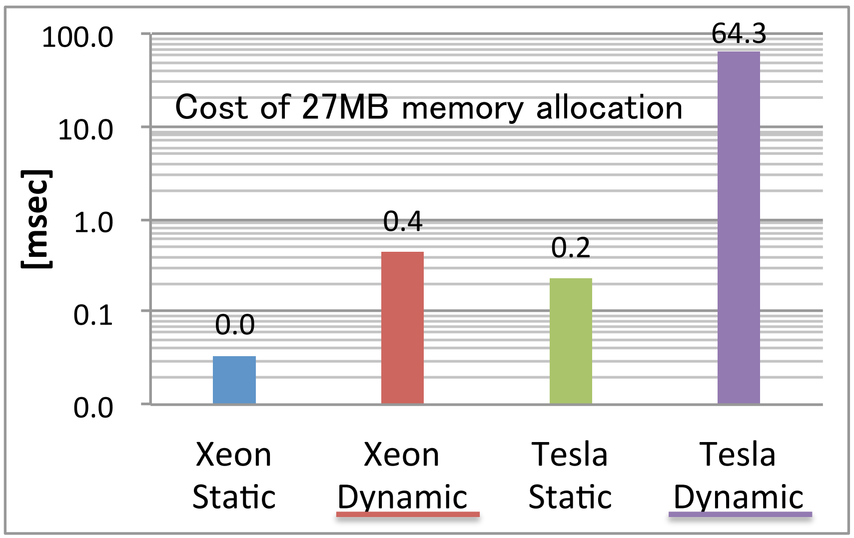
Particle transfer from GPU to CPU mem.

GPGPU server (node spec.)		
	CPU:1 core (Xeon E5607)	GPU:448 cores (Tesla M2075)
Peak perform.	11.72GFLOPS	1030 GFLOPS
Bandwidth	32 Gbyte/sec	150 Gbyte/sec

4.2 GB test data	Serial	Simp. para.
[msec]	66070.4	25000.3

Only 3x speed up using 448 cores

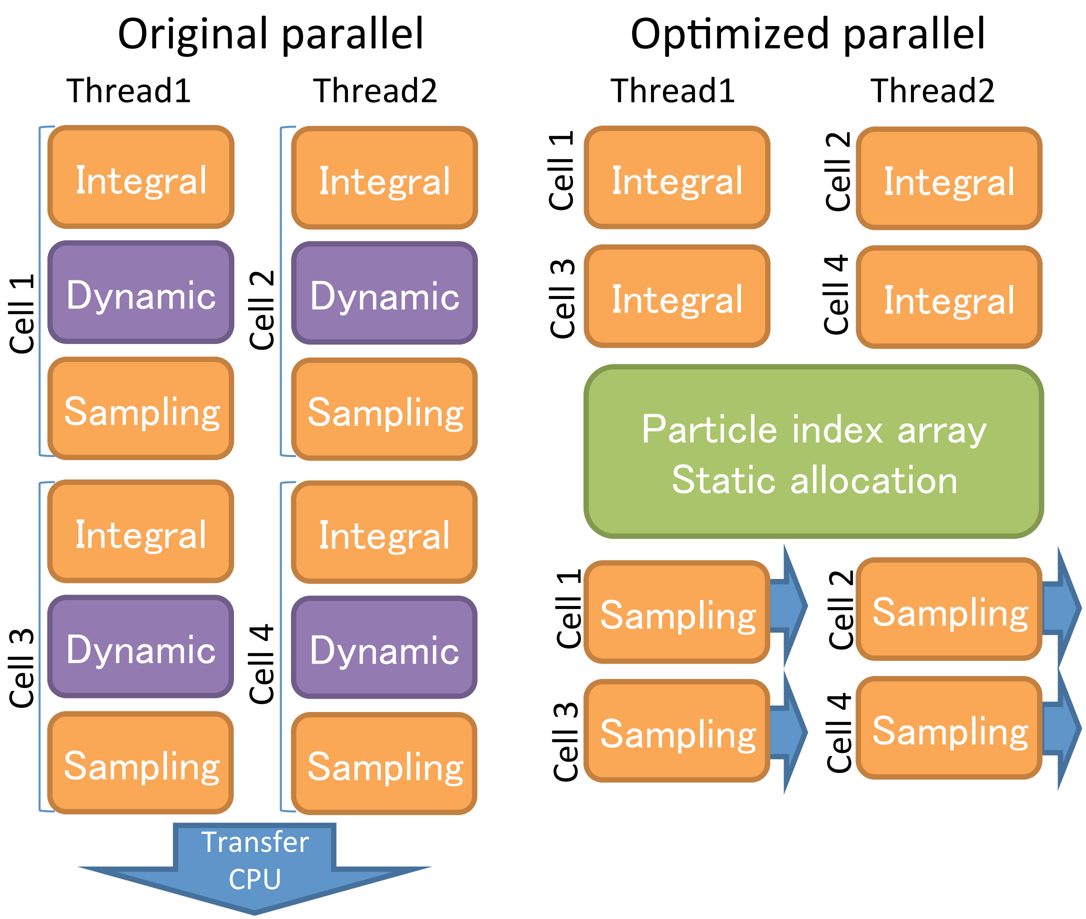
⇨ **Limited by dynamic memory allocation on GPU**



Parallel Particle Generation (GPGPU)[1]

Static allocation of particle array ⟹ Particle index array

- Memory addresses given by cumulative sums of particles at each cell
- Separation of density integral and sampling by memory allocation for particles
- Overlapping particle generation and transfer from GPU to CPU



Processing time of 4.2 GB test data [msec]

Serial	Orig. para.	Opt. para.
66070.4	25000.3	3726.4

Improved parallel accelerated 22 times faster than serial processing and 8 times faster than simply parallel processing.

Dramatic improvement in particle generation speed on GPU.

1. Takuma Kawamura, Yasuhiro Idomura, et al. “Remote Visualization System based on PBVR”, JSST 2014

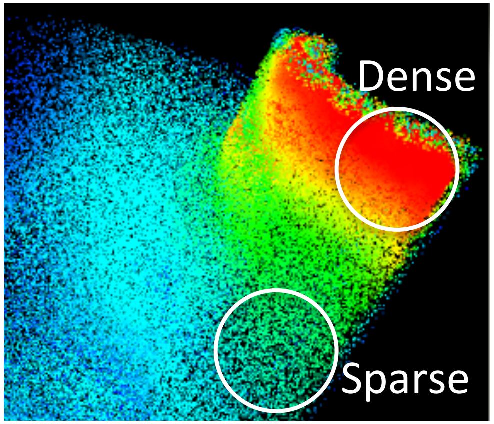
Parallel Particle Generation on Supercomputer

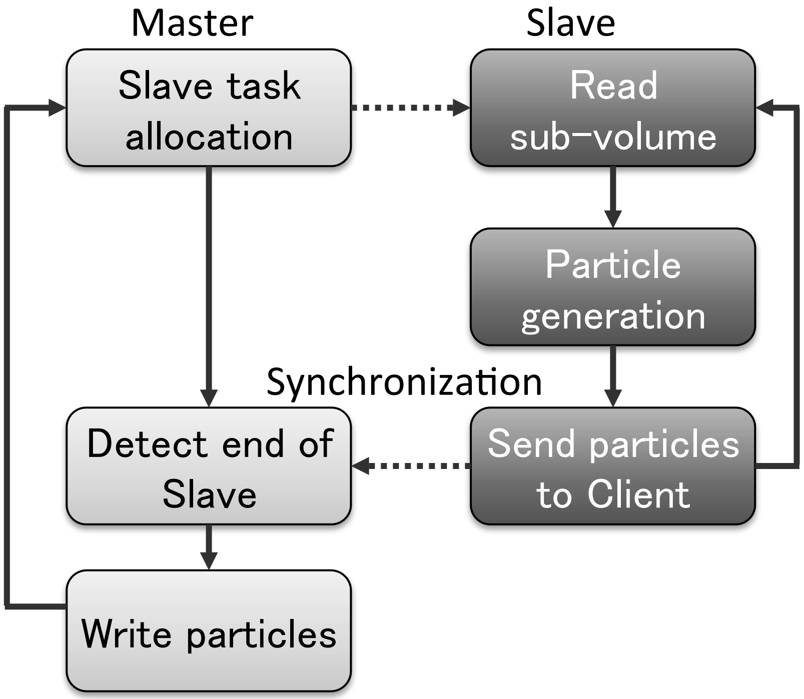
- Accelerate particle generation via massively parallel environment
- Parallelization with hybrid MPI—OpenMP programming model

Non-uniform particle distribution and load imbalance

- Filter
  - Octree decomposition of volume data
- Parallel file I/O of sub-volume data
- Dynamic task allocation
  - Assign sub-volume data to each node using Master-Slave MPI model
  - Random number is generated by massively parallel Mersenne Twister libraly, Kmath Random[1]

Particle distribution





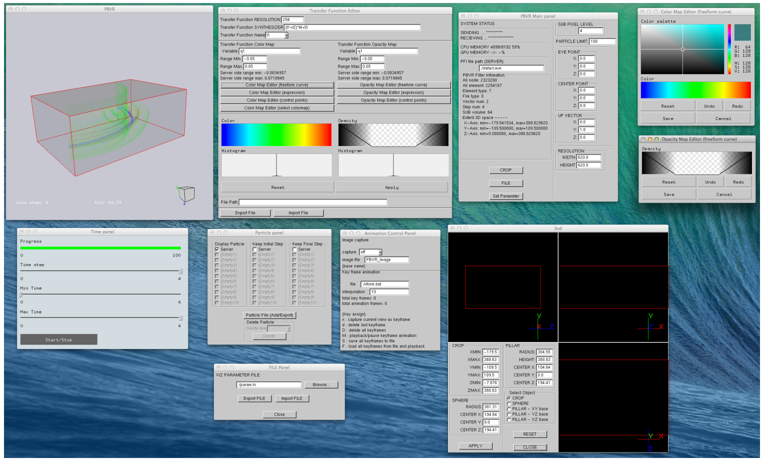
1. Kmath Random [Imamura, AICS, RIKEN]

Numerical Experiment

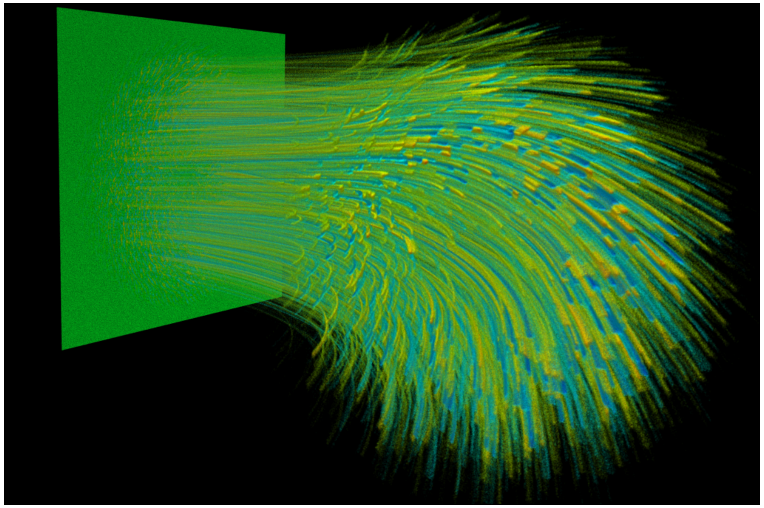
Server machine (node spec.)	
K	SPARC64VIIIfx (8 cores), 128GFLOPS
	6D Torus interconnect
BX900	2 x Xeon x5570 (4 cores), 94GFLOPS
	Infiniband QDR(Fat tree)

The electrostatic potential of turbulent fluctuations in a fusion plasma

Data size and image resolution	
Num. Cells	286×10^6
Data size	1.1GB x 100 step
Resolution	1024x1024
Num. particles	$\sim 10^7$

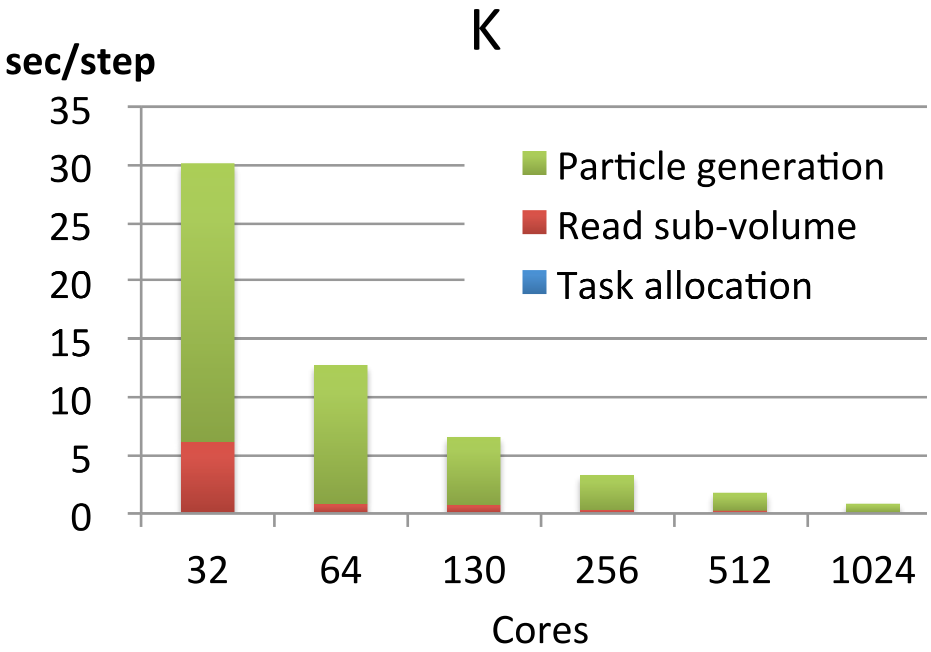


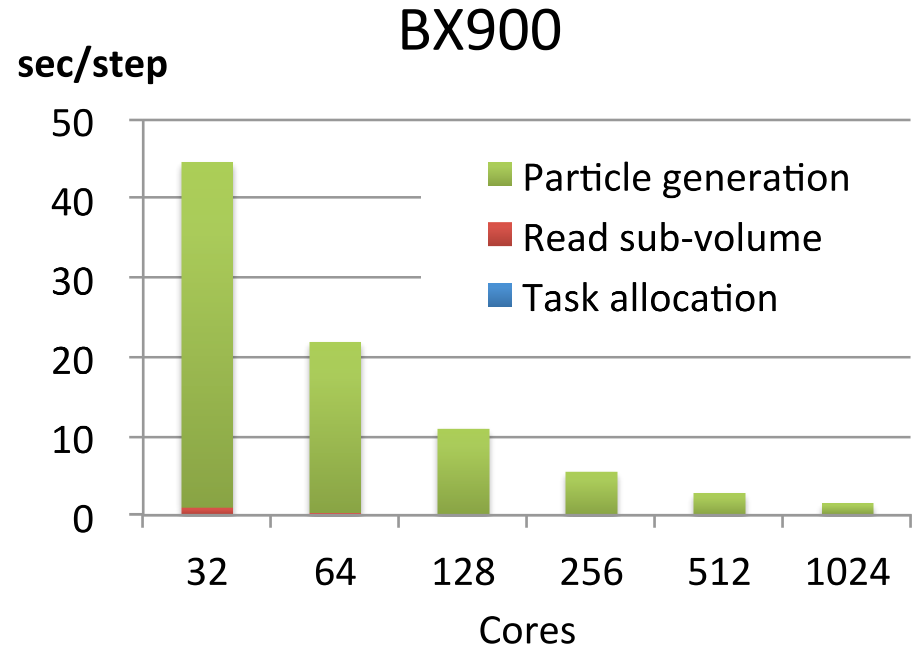
User interface of PBVR



ITER size full-f simulation [Idomura,POP14]

Strong Scaling on Supercomputer[1]





Computational cost of particle generation for 100 step

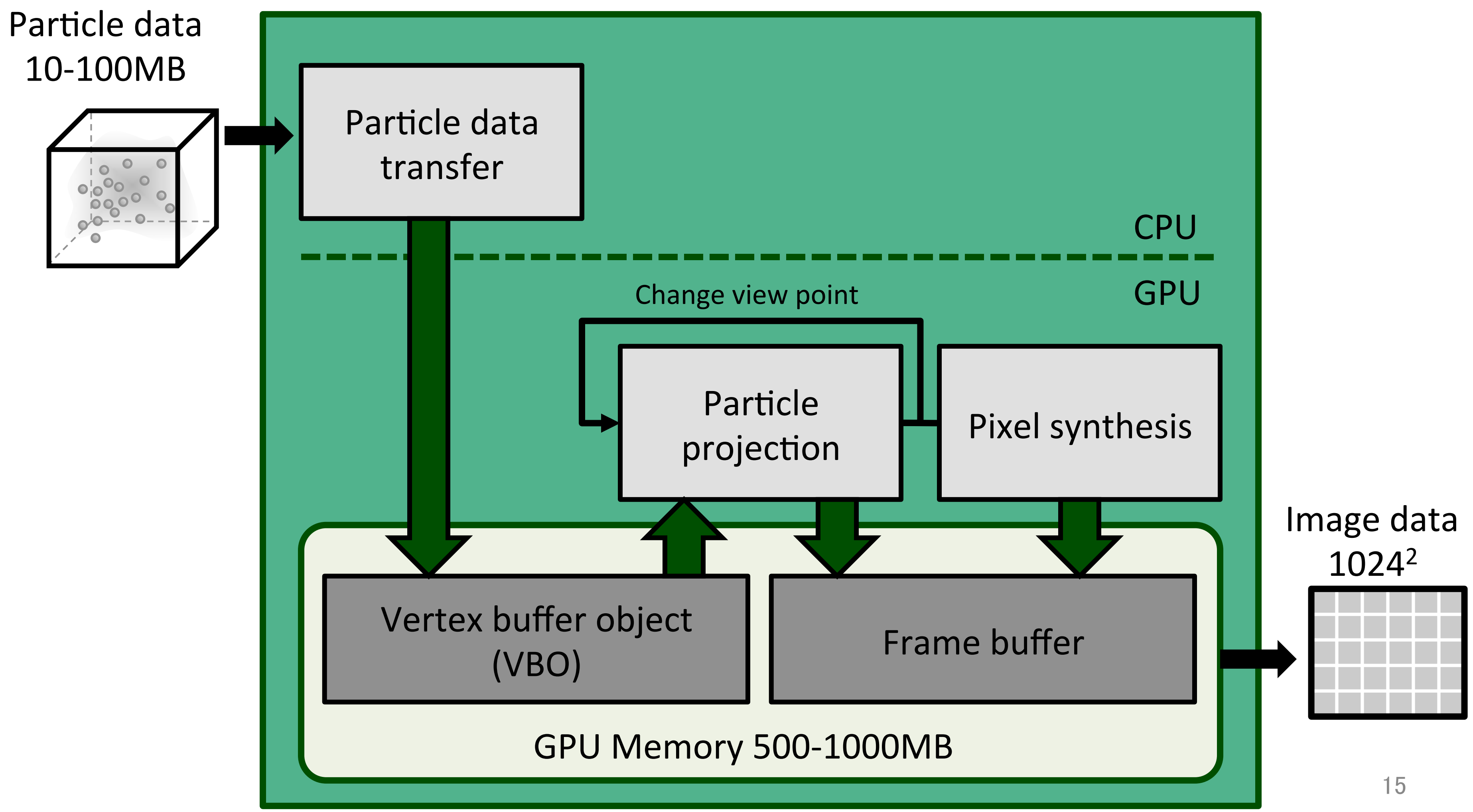
- Strong scaling up to 1024 cores
  - The parallel efficiency is more than 90%
  - 1.36 times faster than BX900
- K computer achieved 0.84 [sec/step] at 1024 cores

1. T. Kawamura, Y. Idomura, H. Miyamura, et al. " Remote Visualization System based on PBVR", VDA 2015

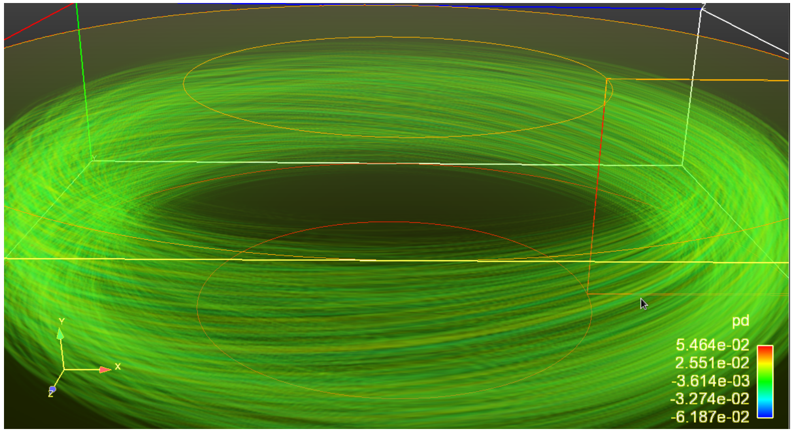
Particle Rendering using GPU

Enabling of interactive rendering using GPU acceleration

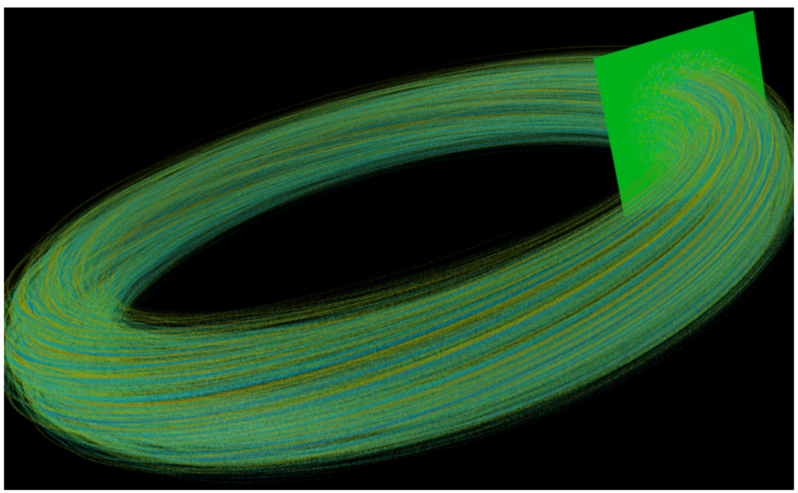
- Frame buffer utilized pixel synthesis
- Small particle data stored on GPU memory



Benchmark Test against the EnSight



EnSight



PBVR

Client-Server Environment		
Client (PC)	CPU	Xeon E5 2690 x 2
	GPU	Quadro K5000
Server1 (CPU)	BX900 (6nodes/48 cores)	
Server2 (GPGPU)	Tesla M2075 (6 nodes)	
Network	3.4 [MB/sec]	

sec/step	Ensight (CPU)	PBVR (CPU)	PBVR (GPU)
Filter [sec/step]	-	0.7	0.7
Image gen. [sec/step]	-	51.4	3.8
Transfer [sec/step]	-	75.7	75.7
Total [sec/step]	3873	127.8	79.5
Frame rates [fps]	2.7	60.0	60.0
Client memory [MB]	900	257	257

- ~30x and ~50x speed up in the total performance on CPU and on GPGPU
- Particle generation on GPGPU is ~13.5x faster than CPU
- ~22x speed up in the frame rates

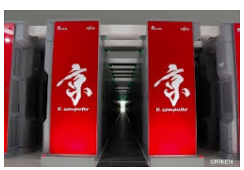
Client-Server Communication

User PC (Lab.,Pocket WiFi, etc…)

Linux
Mac
Windows

ssh tunnel
(Internet)

Supercomputer (K computer, BX900, etc.…)

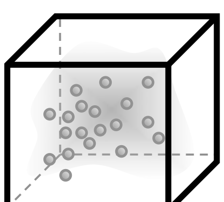


ssh port
forwarding

Calculation node, Analysis node

Login node

Particle data
10-100MB



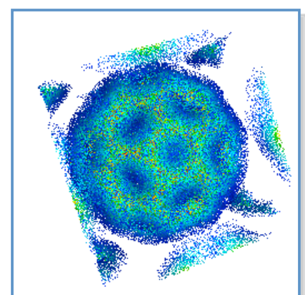
Network bandwidth
1-100MB/sec

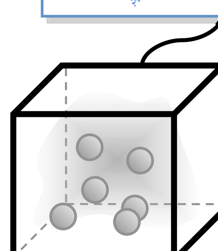
Interactive data exploration

Abstract analysis ⇄ Detail analysis

Find Region of Interest (ROI)
Control of Level of Detail (LOD)

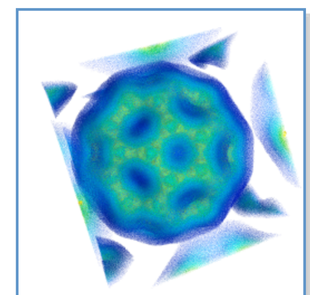
Coarse / Fast
(~1MB particles)

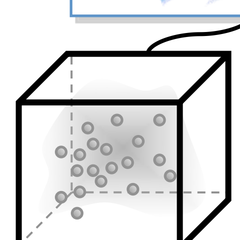




Small number of particles

Fine / Slow
(~100MB particles)





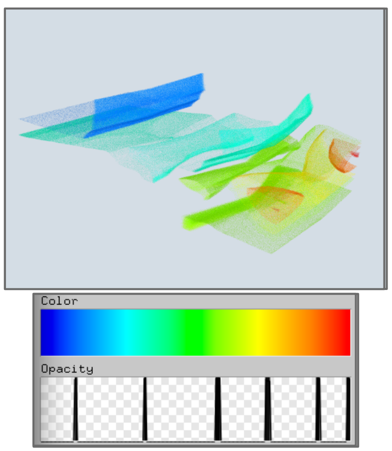
Large number of particles

General Visualization Interface[1]

Important visualization operations

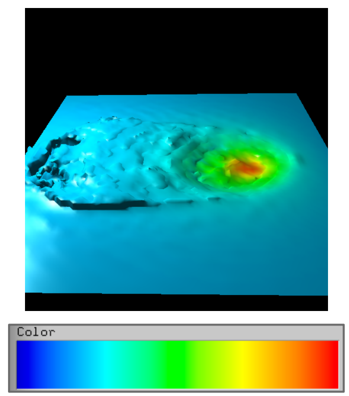
- Shape extraction using iso-surface
- Multi-variables assigned to different properties
- Composition of different visualization methods

⇨ Above operations are feasible by PBVR and integrated transfer function

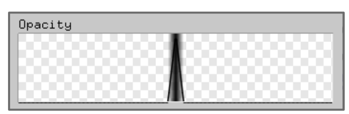


Water pressure response

Iso-surfaces extracted by spiky opacity function

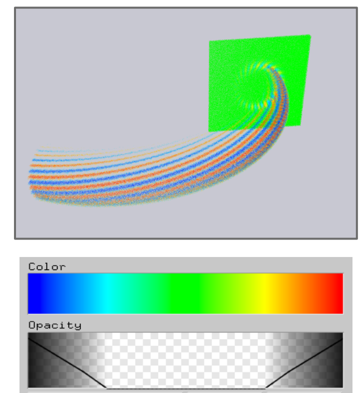


Temperature



Charge density

Two variables assigned to color and opacity



Electrostatic potential

+

Z-coordinate

Composition of two variables and methods

1. T. Kawamura , Y. Idomura , et al. “Multivariate Volume Rendering Using Transfer......”, SIGGRAPH ASIA 2015

Summary

Clinet-Server visualization based on PBVR

- ❖ Sort-less volume rendering using particle data
- ❖ Scalable Monte-Carlo particle generation
- ❖ Transfer small particle data and rendering using graphic card

Interactive remote visualization of extreme scale data

Present status and future issues

- ❖ Open-source software PBVR was released in Mar. 2015
  http://ccse.jaea.go.jp/ja/download/software_eng.html
- ❖ Supported data formats: KVS, AVSFLD/UCD, VTK, PLOT3D, STL
- ❖ Server on massively parallel systems (K, BX900…), GPGPU clusters, and PCs
- ❖ Client on PCs (Linux/Mac/Windows)
- ❖ Advanced transfer function design for multivariate data
- ❖ High dimensional data (5D particle distribution, 4D spatio-temporal data)
- ❖ Multi-scale data

Brief Demo. of PBVR with Client-Server

Connection via Internet

- Client: user PC, Server: K computer
  - Pre-post node of K computer for interactive processing
- Local and remote Port forwarding using ssh tunnel

Visualization

- Nuclear fusion plasma simulation result composed of 4 variables
  - electrostatic potential
  - electron density fluctuation
  - electron temperature fluctuation
  - magnetic potential
- Multivariate transfer function
  - Visualize relation of 4 variables

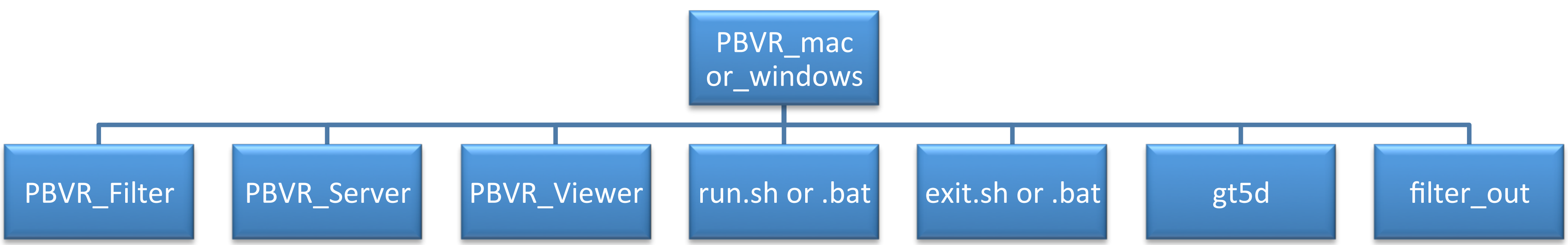
Hands-on with PBVR

Today's Goals

- Install prebuilt binary of PBVR
- Run PBVR system with standalone mode
- Become familiar with viewer window
- Image quality control
- Designing transfer functions
- Saving results
- Example of batch mode

Installation of Prebuilt Load Modules

- Decompress the hands-on package
  - Mac: PBVR_mac.tgz
  - Windows: PBVR_Windows.zip
- Choose the suitable package to your platform and copy it to any working directory
  - Don’t change the directory structure in the package



Package Contents

Prebuilt load modules

- v1.06a contains only serial prebuilt binaries for Mac and Windows
  - Compiled without OpenMP
  - Including static link for various libraries

Start script

- Launching prebuilt load modules

Stop script

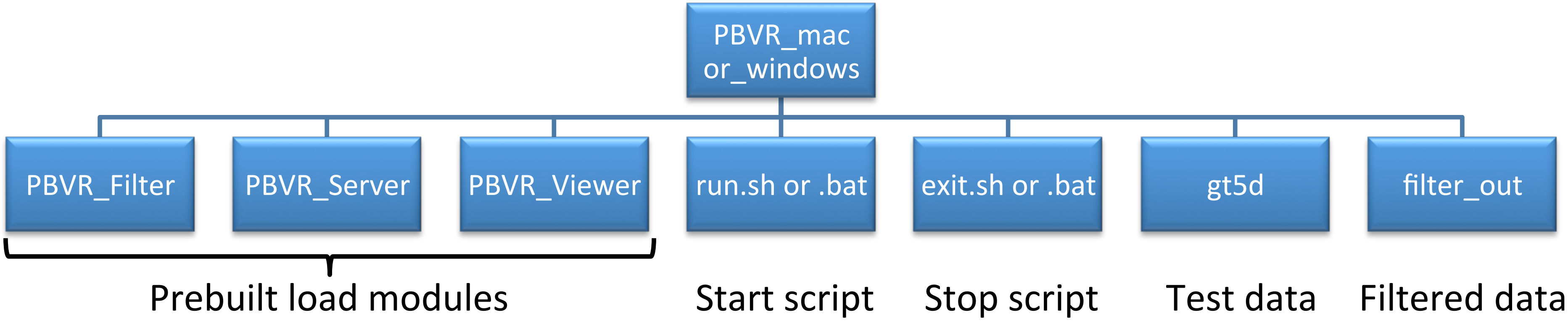
- Terminating PBVR client and server

Test data (gt5d)

- Nuclear fusion plasma simulation result composed of 2 variables
  - Electrostatic potential
  - Magnetic potential

Filterd data (filter_out)

- Store decomposed data by filter



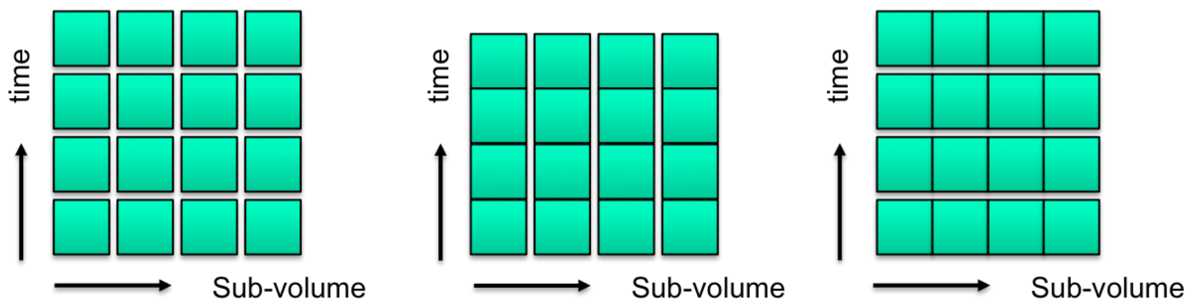
Package Contents

Start script

```
./PBVR_Filter ./gt5d/param.txt
./PBVR_Server &
./PBVR_Viewer -sl 4 -plimit 1000 -pa demo.tf
  -vin ./filter_out/case.pfi -shading P,0.6,0.6,0.6,30
```

PBVR_Filter

- Octree decomposition parameters
  - I/O directories
  - Number of layers
  - Output file format
- Supported data formats
  - KVS, AVSFLD/UCD



SPLIT format | Sub-volume aggregate format | Step aggregate format

PBVR_Server

- Launching in the background

PBVR_Client

- Rendering parameters
  - vin: filtered data
  - pa: transfer function
  - sl: subpixel level
  - plimit: particle limit
  - shading: shading model and parameters

Package Contents

Test data (gt5d)

- **gt5d**
  - **gt5d.fld** — An AVS field file
  - **co3d.dat** — A coordinate data file
  - **pd3d.dat** — The variable 1 (Electrostatic potential)
  - **psid.dat** — The variable2 (Magnetic potential)
  - **param.txt** — Input parameters for PBVR Filter
  - **demo.tf** — A transfer function file for demonstration (Generated by client)

param.txt

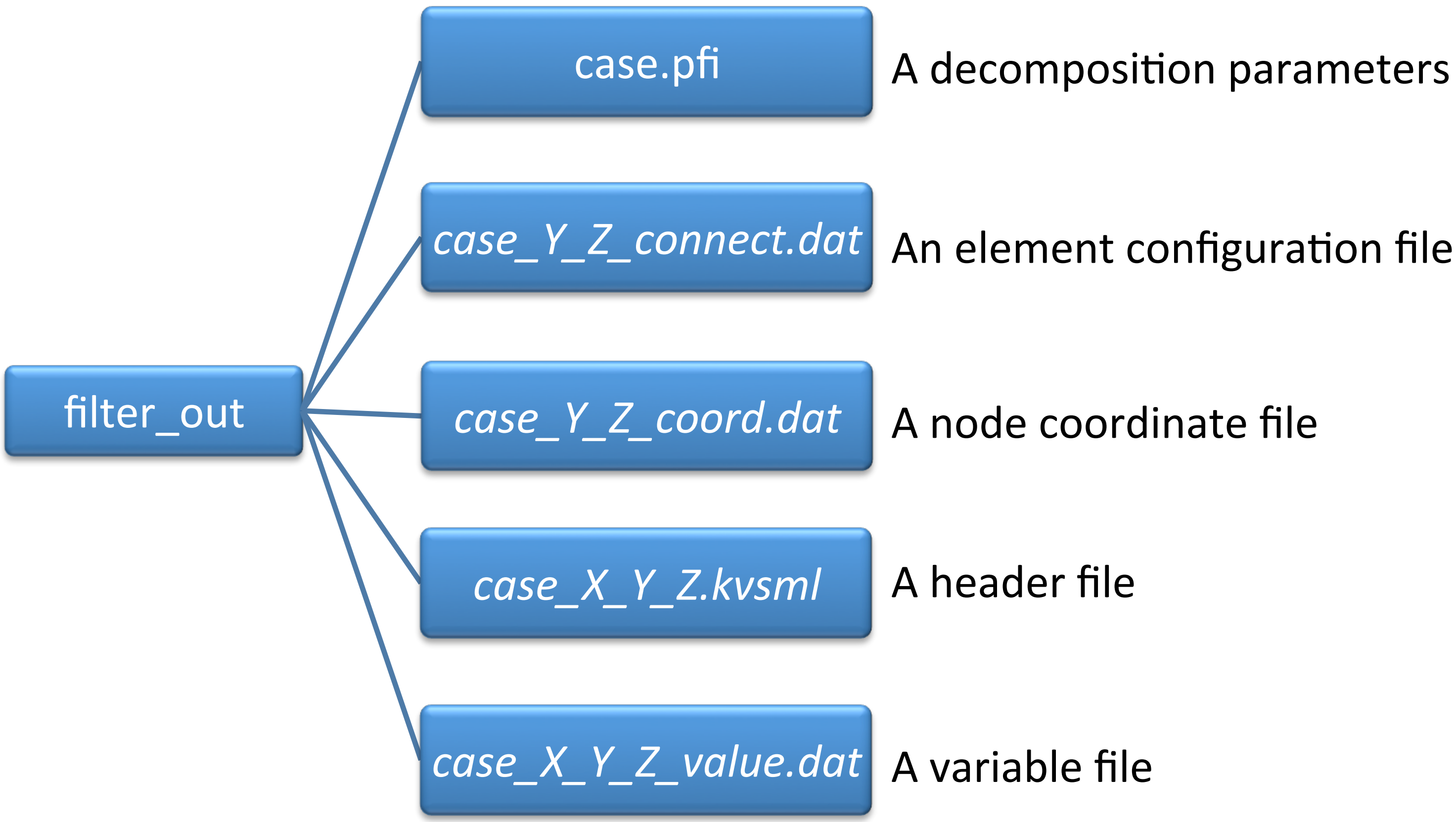
```
in_dir=./gt5d
field_file=gt5d.fld
out_dir=./filter_out
out_prefix=case
start_step=0
end_step=4
```

Omitted default parameters

- SPLIT format
  - output_type=0
- A single sub-volume
  - n_layer=0

Package Contents

Filtered data (filter_out)

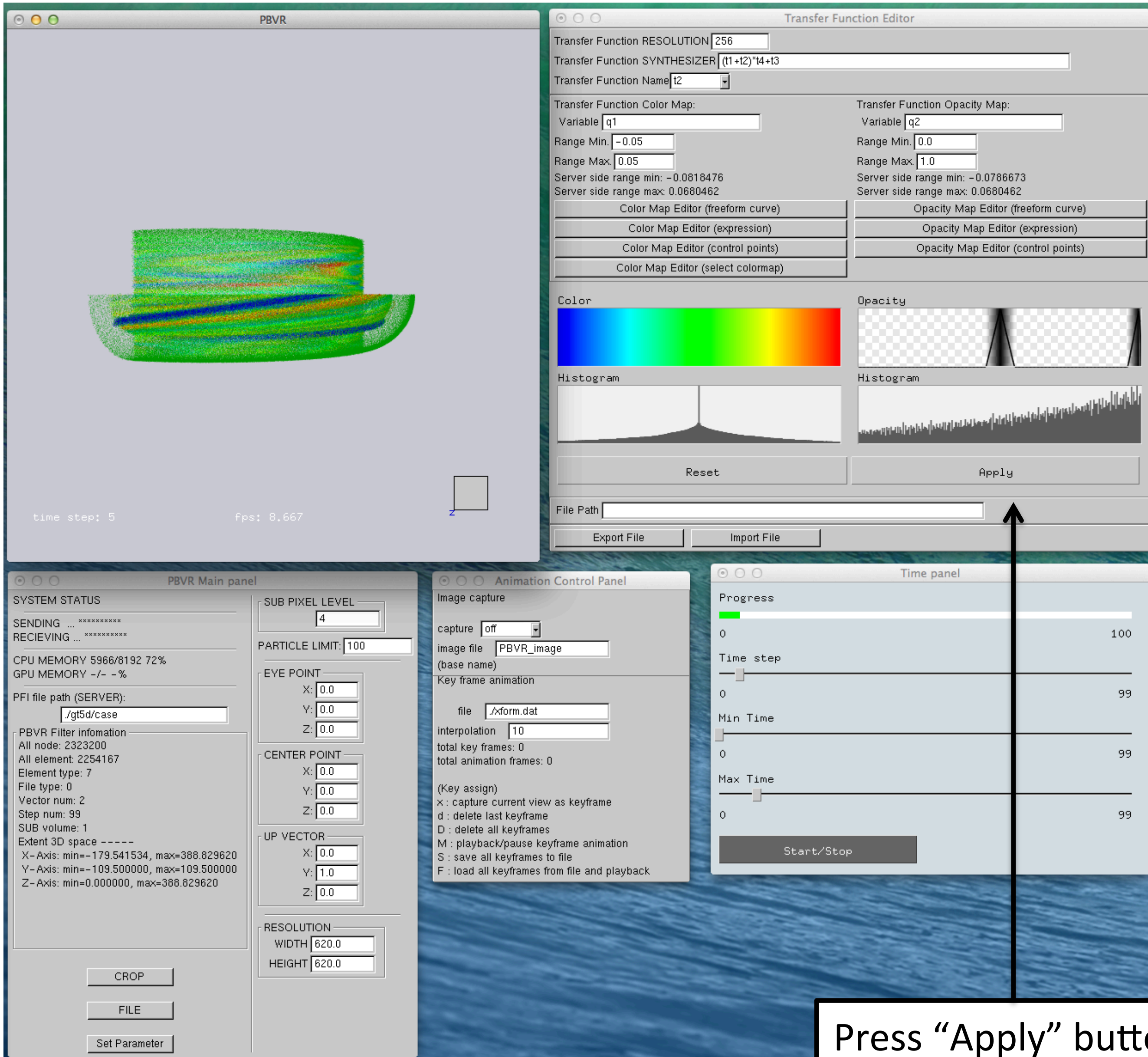


- X : number of steps（in 5 digits）
- Y : index for sub-volume (in 7 digits)
- Z : total number of sub-volumes（in 7 digits）

Launching PBVR Client-Server System

- Windows
  – Execute the batch file "run.bat" by double-clicking
- Mac
  – Execute the script "run.sh" in the terminal

Launching PBVR Client-Server System



Press “Apply” button

Viewer Operations

- Operations
  - Rotation: move the mouse while pressing the left-button
  - Translation: move the mouse while pressing the right-button
  - Zoom: scroll up/down the mouse wheel, or move the mouse up/down while pressing the **Ctrl** key
  - Reset position: home button (fn + left arrow on Mac)
  - Changing rotation target: **l** key → light, **o** key → object

- Display
  - **time step**: current time step of the data
  - **fps**: the frame rate [frame/sec]
  - Orientation box: x-y-z axes

Try viewer operations

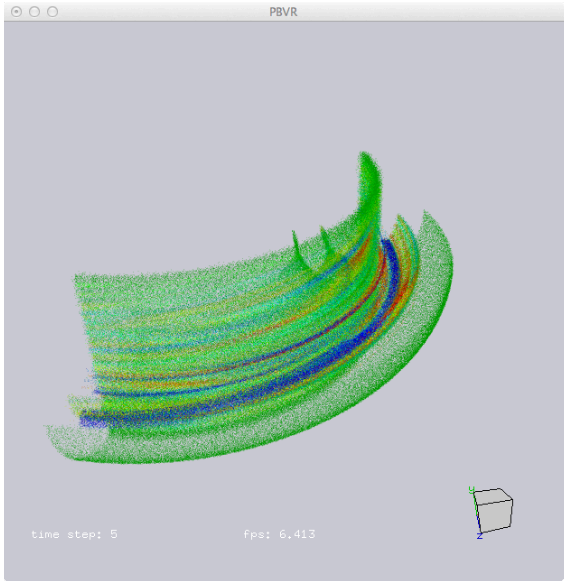
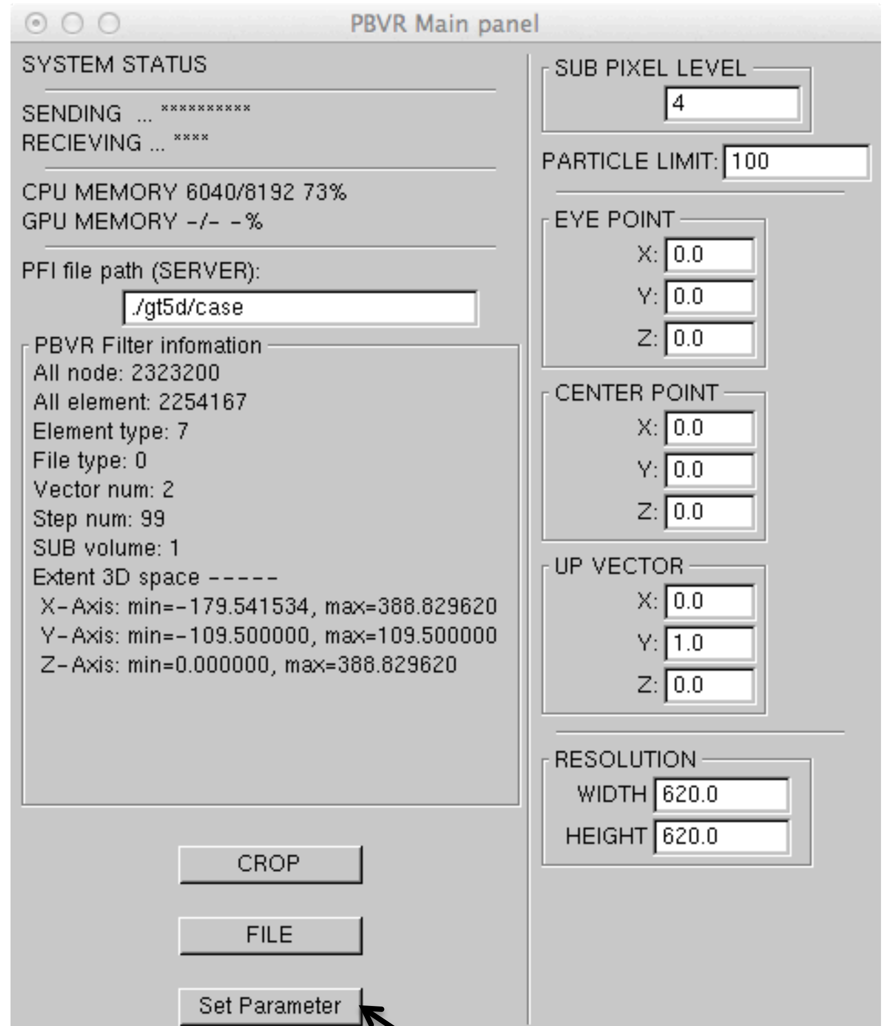


Image Quality Control

Change visualization parameters

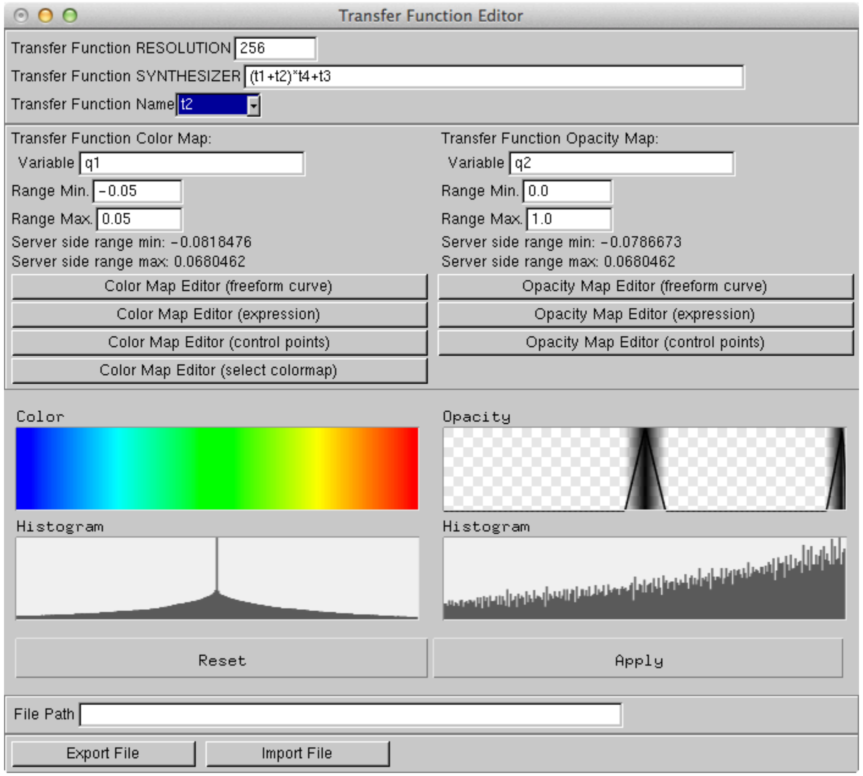
- **SUB PIXEL LEVEL**
  - Division number of pixel
  - Proportional to image quality and number of particles
- **PARTICLE LIMIT**
  - Maximum number of particles
  - The number multiplied by 10^6.
- **RESOLUTION**
  - Viewer's resolution.



Press "Set Parameter" button after setting

Designing Transfer Functions

- **Transfer Function Editor** provides a multi-dimensional transfer function design
  - Two independent variable quantities to color and opacity
  - Definition of each variable with an arbitrary function
    - *X-Y-Z* coordinates
    - variables *q1*, *q2*, *q3*…
  - Multidimensional transfer function
    - Synthesis of one-dimensional transfer functions *t1*-*t5*



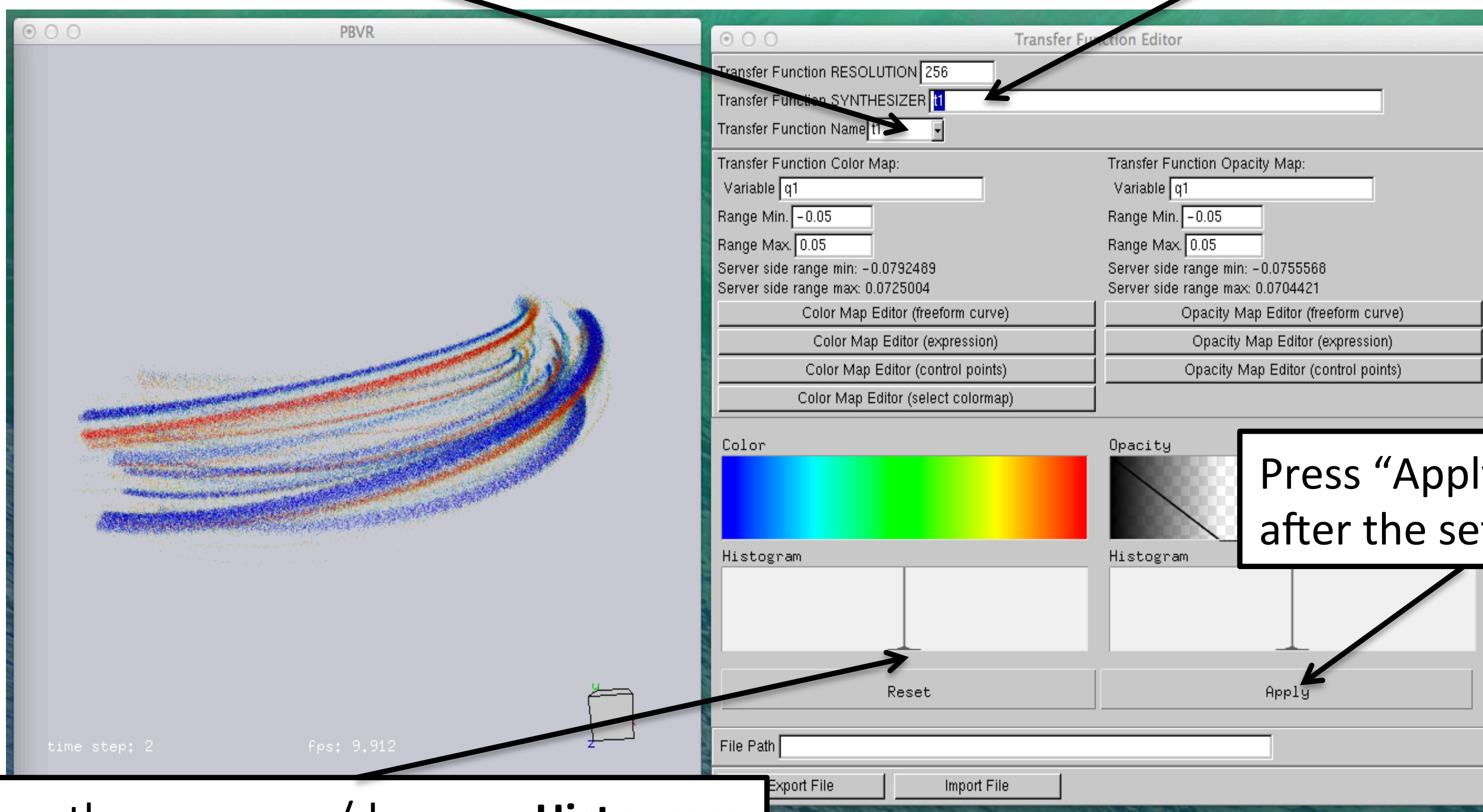
Designing Transfer Functions

Volume Rendering for a Single Variable

- variable *q1* by setting the transfer function *t1*

Select “t1” from **Transfer Function Name**

Input “t1” to **Transfer Function SYNTHESIZER**



Press “Apply” button after the settings

Drag the mouse up/down on **Histogram**

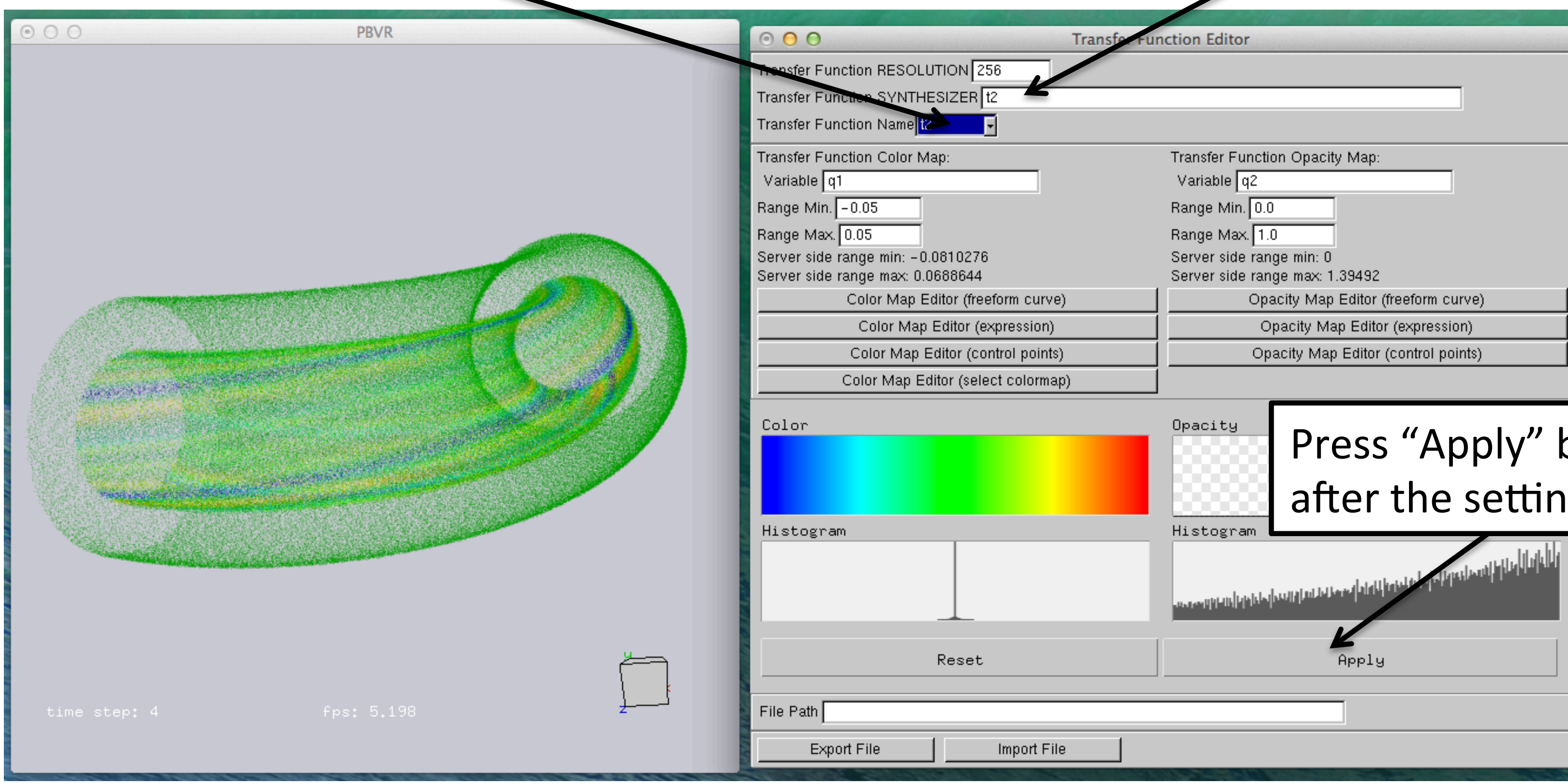
Designing Transfer Functions

Multivariate Volume Rendering

- Colors assigned to the variable *q1*
- Opacities assigned to the variable *q2*.

Select "t2" from **Transfer Function Name**

Input "t2" to **Transfer Function SYNTHESIZER**



Press "Apply" button after the settings

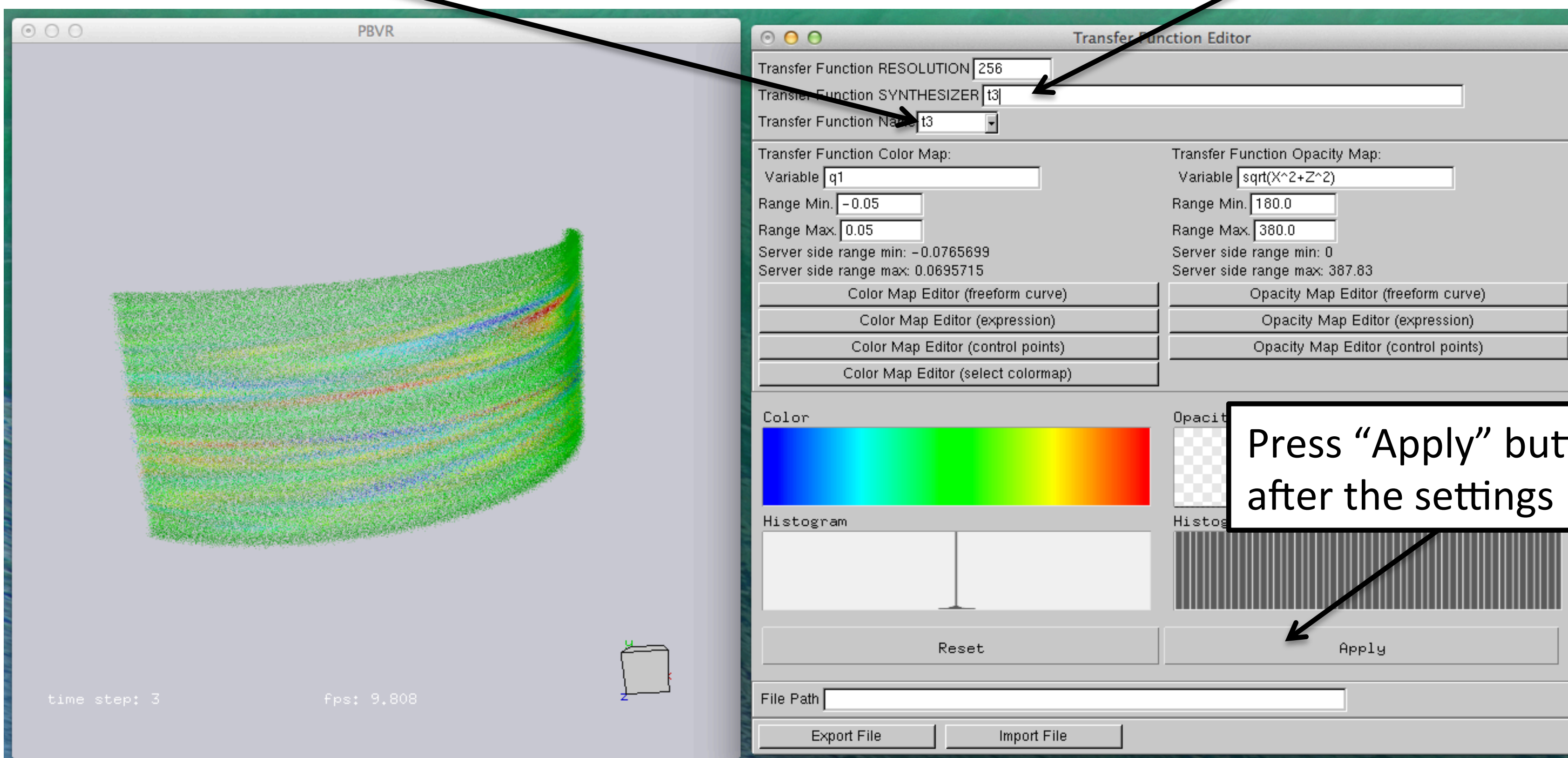
Designing Transfer Functions

Extracting a Slice

- Colors assigned to the variable *q1*
- Cylindrical surface (*X^2+Z^2=const.*) as the variable of opacity

Select "t3" from **Transfer Function Name**

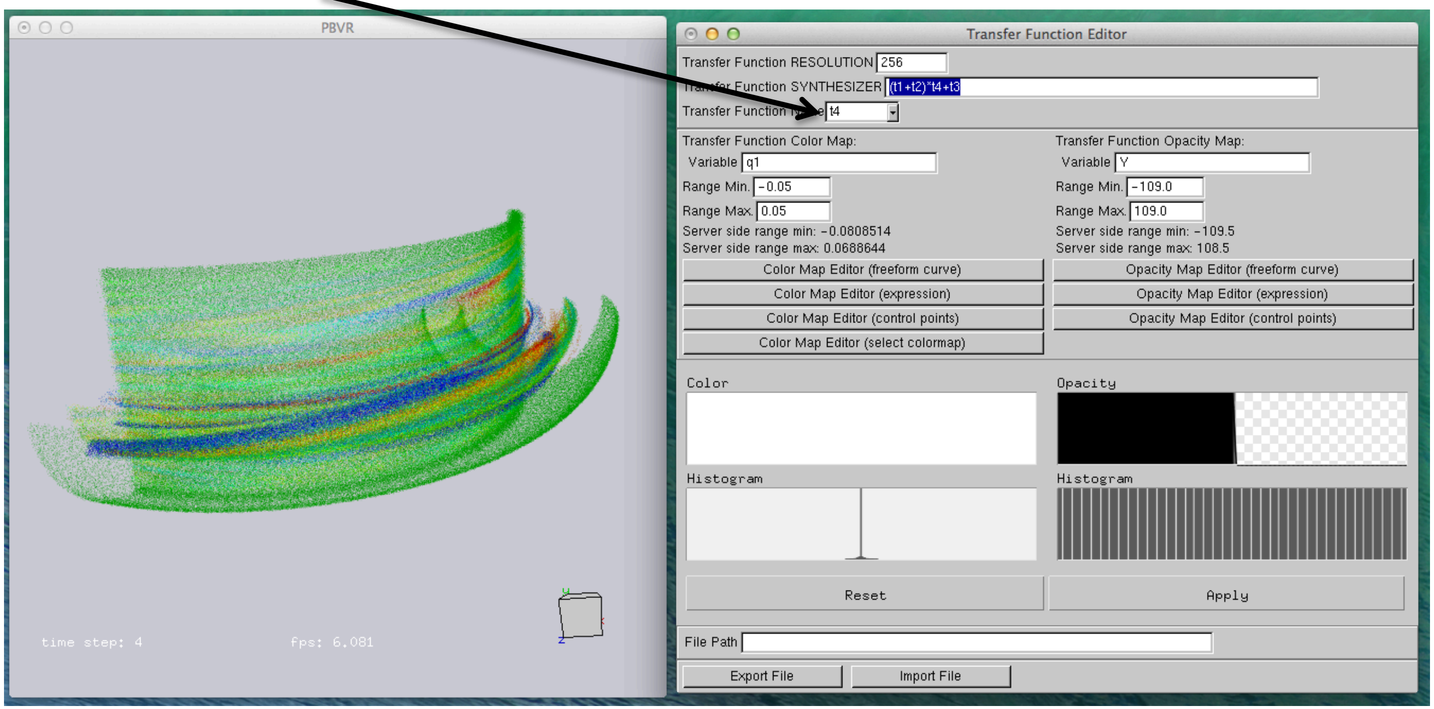
Input "t3" to **Transfer Function SYNTHESIZER**



Synthesis of Transfer Functions

- Transfer function *t4* makes the region *Y* > 0 transparent.

Select “t4” from **Transfer Function Name**

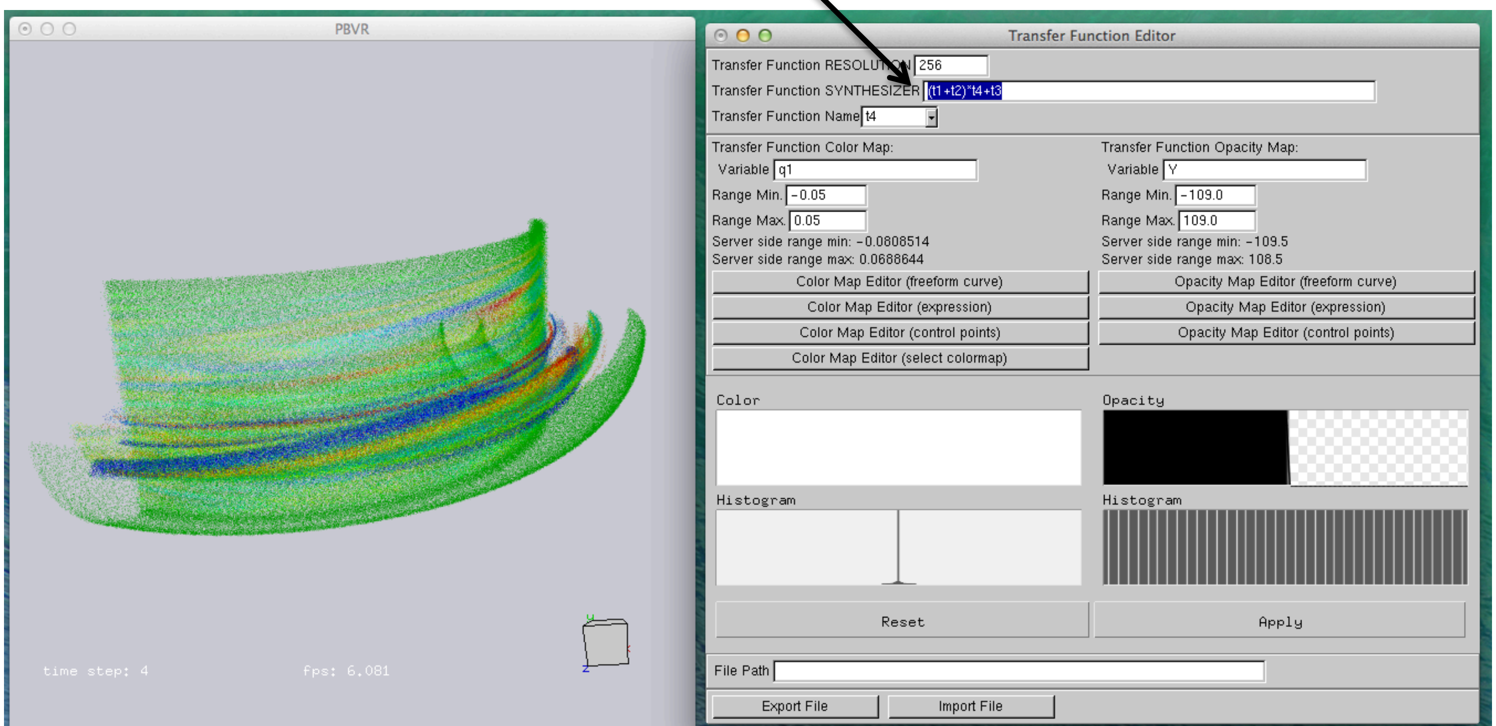


Designing Transfer Functions

Synthesis of Transfer Functions

- Flexible composition through arithmetic operations
  - synthesizing *t1, t2, t3* and *t4* as ***(t1 + t2) * t4 + t3***

Input “(t1 + t2) * t4 + t3” to **Transfer Function SYNTHESIZER**



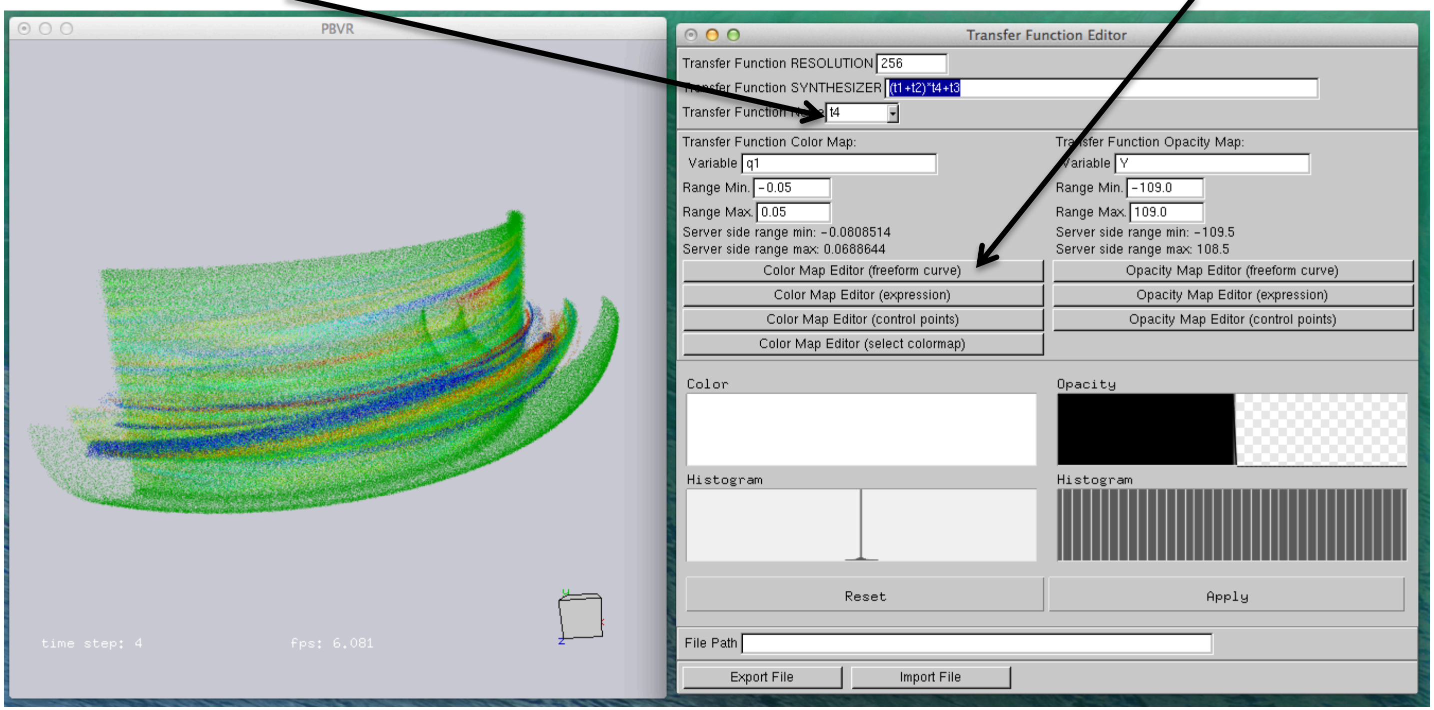
Designing Transfer Functions

Synthesis of Transfer Functions

- Colors of *t2* and *t3* set to (*R*, *G*, *B*) = (0, 0, 0)
- Color of *t4* set to (*R*, *G*, *B*) = (1, 1, 1)

Select “t2/t3/t4” from **Transfer Function Name**

Press “**Color Map Editor (freeform curve)**” button

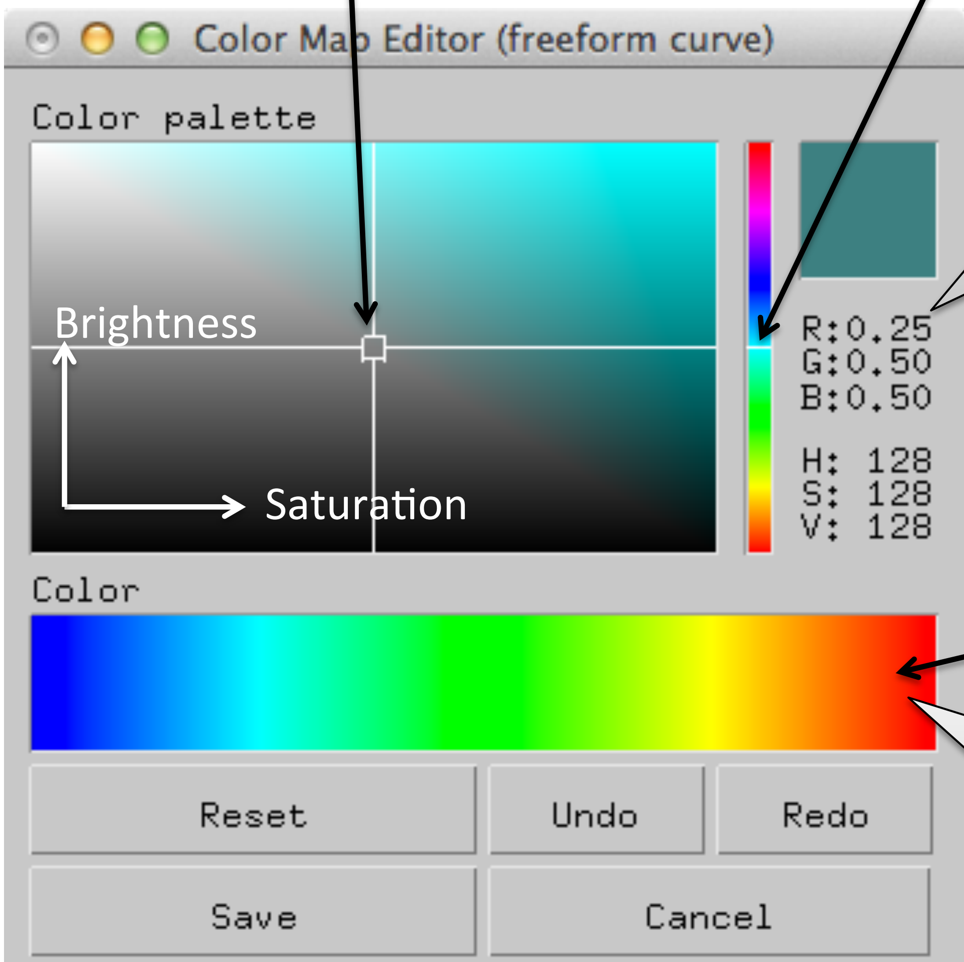


Designing Transfer Functions

Color Map Editor (freeform curve)

Pick up saturation and brightness

Pick up color hue

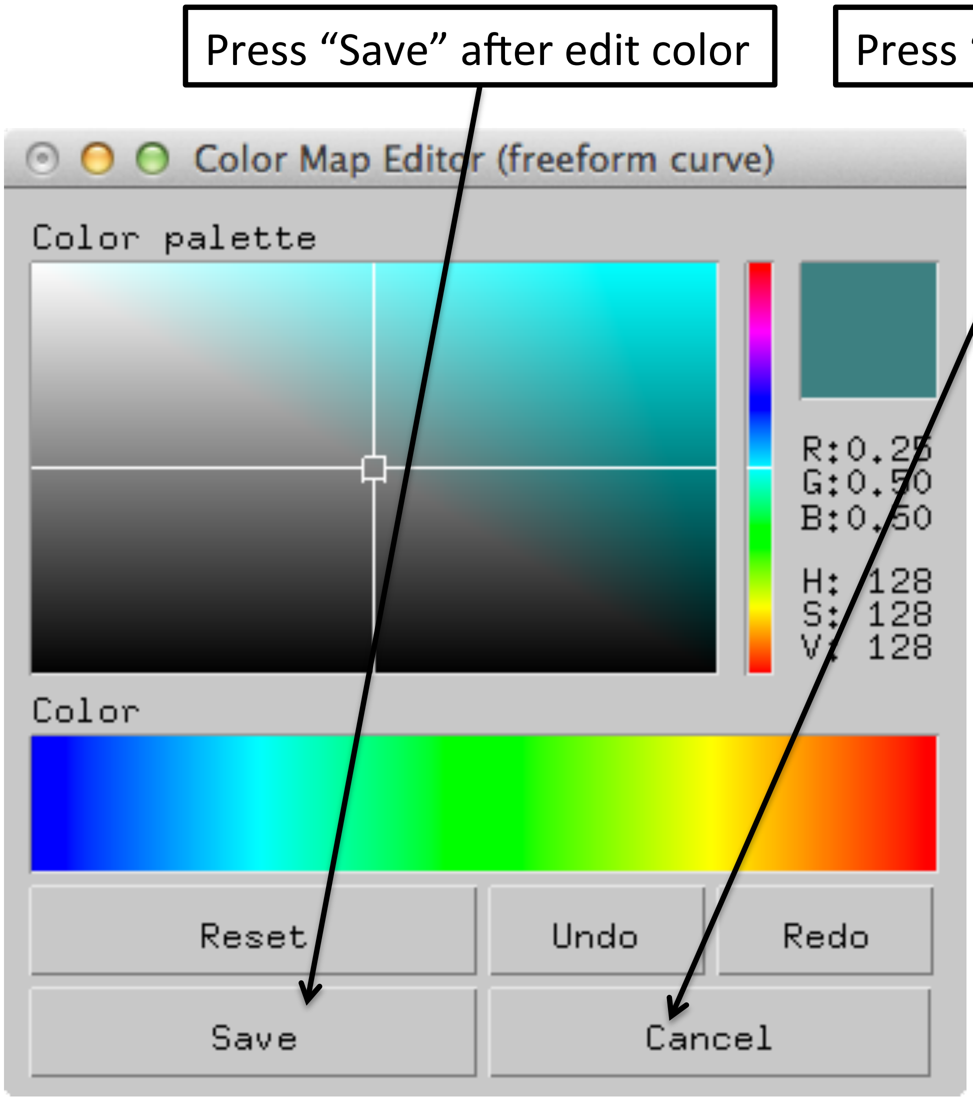


(*R*, *G*, *B*) color normalized to [0,1]
(*R*, *G*, *B*) = (0, 0, 0) means black
(*R*, *G*, *B*) = (1, 1, 1) means white

Overpaint the color by dragging the mouse cursor while pressing the left mouse button

The blending ratio of the original color and the overpainting color is determined by the mouse cursor's vertical position

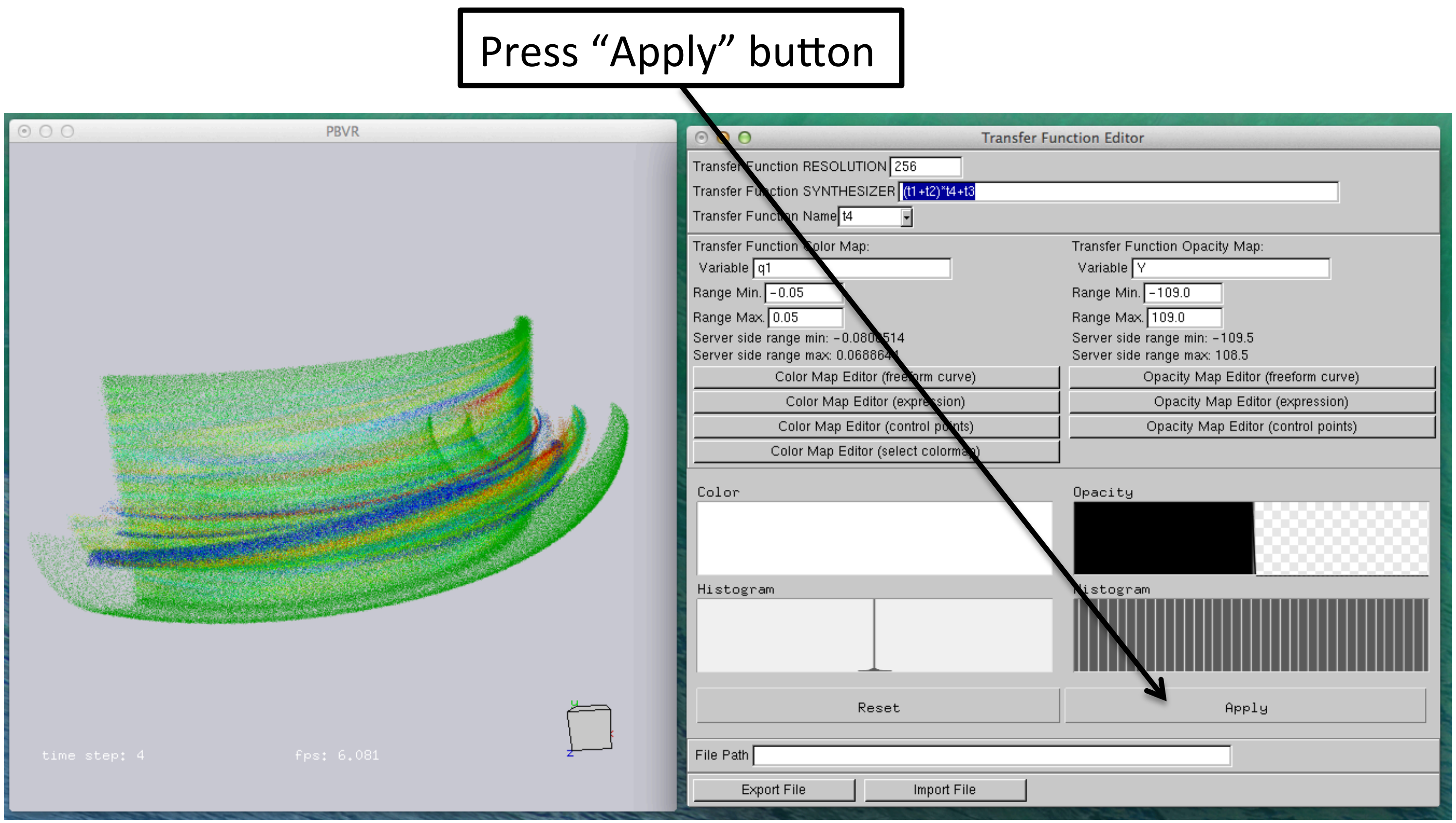
Color Map Editor (freeform curve)



- **Reset**
  – Resets the panel.
- **Undo**
  – Undoes the last mouse action.
- **Redo**
  – Redoes the last mouse action undone.
- **Save**
  – Saves the transfer function.
- **Cancel**
  – Closes the panel.

Synthesis of Transfer Functions

- Final colors obey the colormap defined for *t1*
- In case of red: $(\boldsymbol{R_{t1}} + \boldsymbol{R_{t2}})*\boldsymbol{R_{t4}} + \boldsymbol{R_{t3}}$ where $R_{t2}, R_{t3}=0$, $R_{t4}=1$.

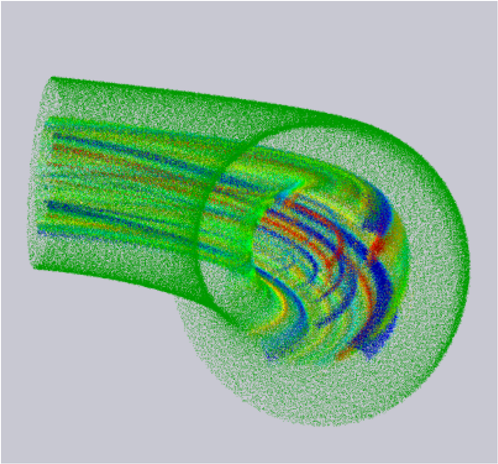


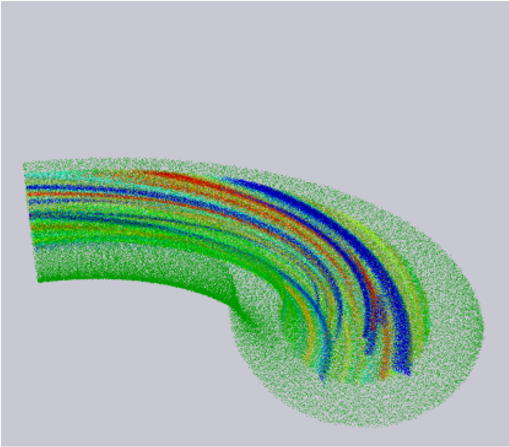
Designing Transfer Functions

Synthesis of Transfer Functions

- ***(t1 + t2)***
  - Composition of *t1*'s volume rendering and *t2*'s torus surface
- ***(t1 + t2) * t4***
  - Extracting the lower half region ($Y < 0$) of *t1* and *t2*
- ***(t1 + t2) * t4 + t3***
  - Composition of cylindrical surface given by *t3*

Try above synthesis

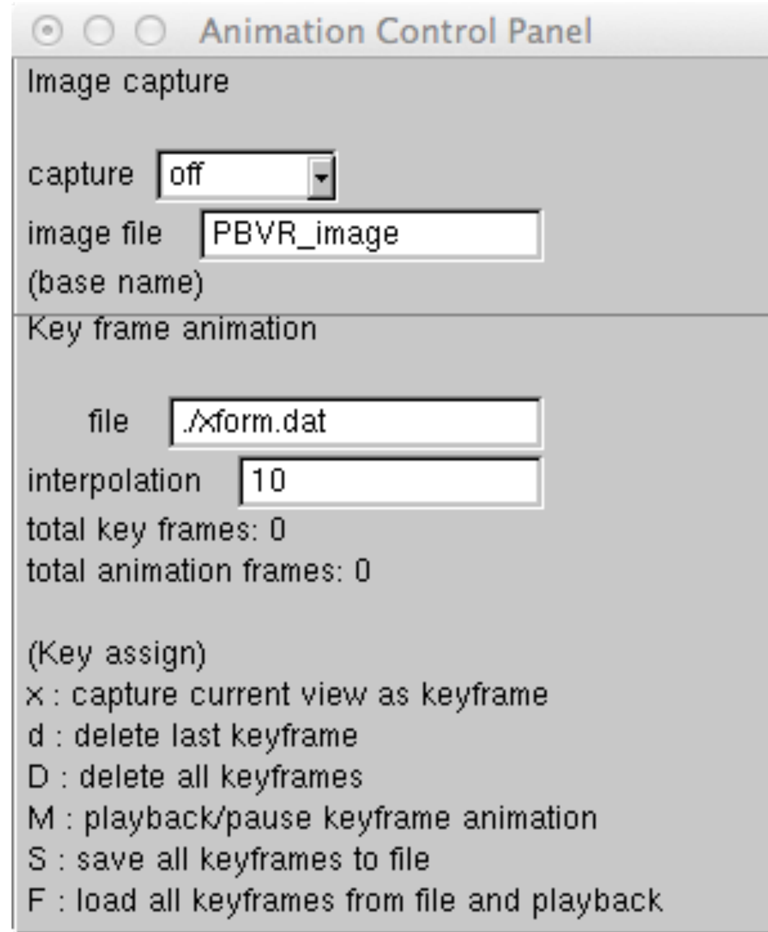




Saving Results

Image Output

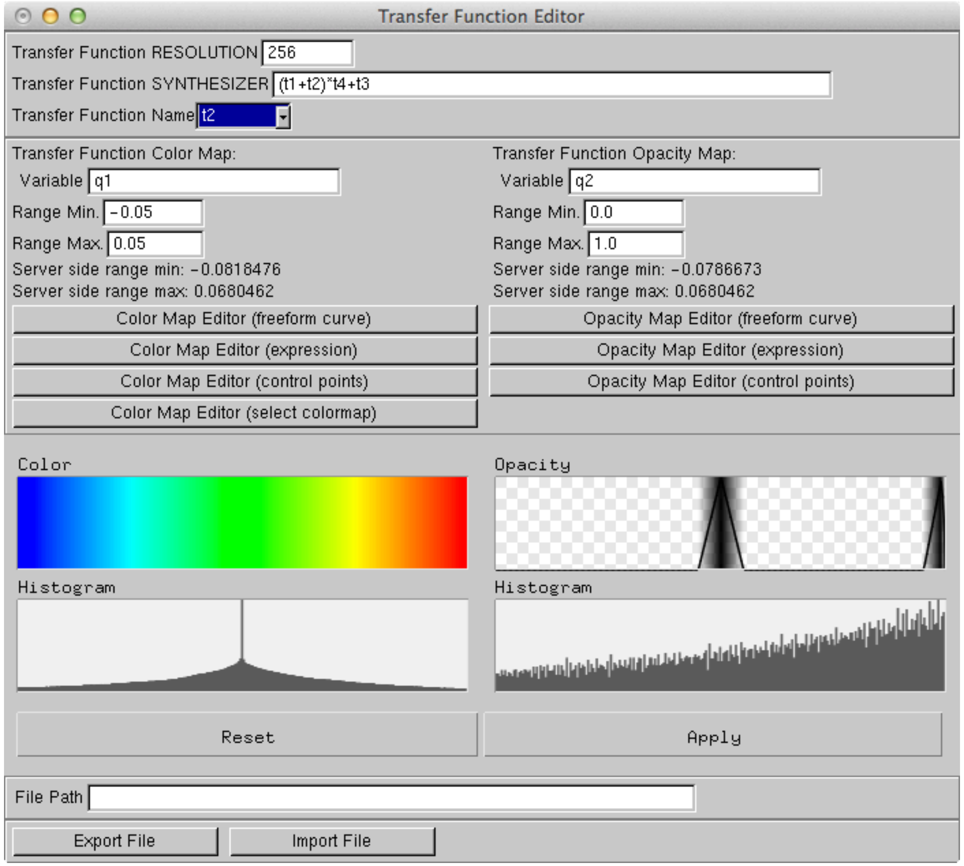
- **capture**
  - Controls on/off of image production
  - 1 shot per time step
- **image file**
  - A prefix of image data files
- **s** key
  - Capturing a snapshot



Saving Results

Transfer Function File

- **File Path**
  - Specifies a file path for saving and loading a transfer function file.
- **Export File**
  - Saves a transfer function defined with this panel to a file in the same format as the parameter file specified with the command line option '-pa'.
- **Import File**
  - Loads a transfer function stored in a file to this panel



Example of Batch Mode

Server

```
./PBVR_Server -B -vin ./filter_out/case.pfi -pa demo.tf -sl 4
-plimit 100 -pout ./output/case
```

Client

```
./PBVR_Viewer -pin1 ./output/case -sl 4
```

- Useful also for high speed rendering of time series data with PBVR Client in stand-alone mode
- Sub-pixel should be specified also when launching PBVR Client in stand-alone mode

Acknowledgement

- The benchmark tests were performed on BX900 @ JAEA and K-computer @ Riken.
- This work is supported by the MEXT, grant for HPCI Strategic Program Field No.4.
- The client-sever type remote visualization software PBVR developed based on the KVS library, which are developed at Koyamada laboratory in the Kyoto University.